HEIG-VD

Projet droiDynamics

Intelligent Wake On Lan

Cours : IFC2

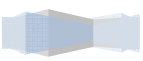
CRINITI Carlo et PIMPAO Claudio

11/01/2010

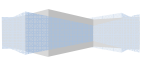
Rapport du projet conçu dans le cadre du module IFC2 à l'Haute Ecole d'Ingénierie et de Gestion du canton de Vaud.

Table des matières

1. Introduction 4
2. Objectifs 5
3. Etat de l'art 5
4. Décomposition du projet 6
5. Technologies 7
5.1. Partie serveur sur Internet 7
5.1.1. SHA512 7
5.1.2. HTML 10
5.1.3. CSS 11
5.1.4. JavaScript 13
5.1.5. PHP 13
5.1.6. MySQL 14
5.1.7. XML 15
5.1.8. DTD 15
5.1.9. Google Maps API 17
5.2. Partie Android 18
5.2.1. Android 18
5.2.2. Open Handset Alliance 20
5.2.3. Licence 20
5.2.4. Versions 20
5.2.5. Architecture logicielle 21
5.2.6. SDK 22
5.2.7. Composants d'une application 24
5.2.8. Intent 28
5.2.9. Appareils présents sur le marché 29
6. Principe général de fonctionnement 31
6.1. Serveur Web 31
6.1.1. Fonctionnement du site internet 31
6.1.2. Méthode d'authentification 35
6.1.3. Echanges d'information web => Android 37
6.2. Android 41
6.2.1. Activités et Service 41
6.2.2. Wake On Lan 46



7. Le code 47
7.1. Serveur Web 47
7.1.1. Base de données 47
7.1.2. Site complet 48
7.1.3. Site mobile 51
7.2. Partie Android 52
7.2.1. Partie concernant l'activité d'amorce 52
7.2.2. Partie Service de l'application : 61
8. Tests 67
8.1. Serveur Web 67
8.1.1. Site Web 67
8.1.2. Site mobile 67
8.2. Android 67
9. Valorisation 69
10. Améliorations possibles 70
10.1. Serveur Web 70
10.1.1. Site Web 70
10.1.2. Site mobile 70
10.2. Android 70
11. Conclusion 71
12. Bibliographie 72
12.1. Serveur Web 72
12.2. Android 72
13. Table des illustrations 73



1. Introduction

Ce projet est réalisé dans le cadre du module d'IFC2 du 1ème semestre de 3ème année de la Haute Ecole d'Ingénierie et de Gestion du canton de Vaud. Il a pour but d'acquérir de l'expérience dans la conception complète d'un projet lié aux technologies mobiles.

Nous sommes partis dans l'idée de toucher le plus de domaines et technologies possibles tout en respectant le temps qui nous était donné. Nous avons touché au cryptage, aux bases de données, aux technologies et langages Web tels que PHP, HTML ou encore JavaScript, aux téléphones mobiles Android ainsi qu'à la programmation Java faisant appel aux possibilités du SDK qui est fourni par Google.

Nous avons donc réfléchit longuement à la façon d'intégrer tous ses éléments dans notre projet pour créer un outil efficace et le plus simple et intuitif possible.

Ce projet s'est déroulé en groupe de deux personnes sur une durée d'un semestre. Dans ce rapport, nous allons donc vous exposer notre travail de manière détaillée. Cependant, il n'est pas utile de s'attarder à expliquer toutes les lignes de code que contiennent nos programmes. Nous allons plutôt vous montrer le fonctionnement de nos classes et la structure de notre programme.

Comme demandé, notre code est complètement commenté et nous avons mis à disposition des utilisateurs un mode d'emploi.

Nous remercions notre professeur et assistant du cours d'IFC2 pour leur encadrement et aide tout au long de ce projet.

Carlo Criniti et Claudio Pimpao

2. Objectifs

Ce projet a consisté à créer un programme pour gérer intelligemment le Wake On Lan. Intelligemment, car il a fallu proposer des fonctionnalités dépendantes de la position de l'utilisateur et de la manière dont il souhaite se servir du programme.

Plus précisément, l'utilisateur va pouvoir créer et personnaliser des points d'intérêt sur une carte où il pourra y ajouter une liste d'ordinateurs à allumer. Par exemple, il pourra ajouter sa maison et son chalet de vacances et y définir tous les ordinateurs qu'il veut pouvoir démarrer en Wake On Lan. Ensuite, en fonction de l'endroit géographique où il se trouve, il aura la possibilité de démarrer ses ordinateurs facilement depuis son téléphone portable.

Voici les points que ce projet implémente :

- Création d'un compte
- Définition et suppression des points d'intérêt
- Définition et suppression de chaque ordinateur à démarrer par point d'intérêt
- Géolocalisation de la personne via son téléphone portable
- Liste des points d'intérêts à proximité de la personne pour démarrer les ordinateurs
- Wake On Lan d'ordinateur

Nous avons décidé de nous concentrer sur la plateforme Android pour le programme sur le téléphone portable. Les raisons de ce choix sont expliquées dans la partie technologies de ce rapport.

3. Etat de l'art

Sur la plateforme Android, il existe plusieurs programmes de Wake On Lan, mais à l'heure actuelle, aucun n'implémente la fonction de géolocalisation comme nous le faisons.

Voici le programme le plus connu qui est présent sur le market[1] Android :

WAKE ON LAN

Figure 1 : Icône de l'application[2]

Développé par : mafro
Version actuelle : 4
Prix : gratuit
Permissions demandées par le programme sur le système : android.permission.INTERNET[3]
Fonctions : - Wake On Lan en précisant l'adresse mac, l'adresse ip et le port
- Historique des anciens profils

[1] Le « market » signifie le marché des logiciels sur Android, une sorte de magasin en ligne où l'utilisateur peut rechercher et installer des logiciels gratuits ou payants.
[2] http://www.androlib.com/appicon/app-qCt.cs.png
[3] Il faut savoir que chaque application étant développée pour Android doit avertir quelles fonctionnalités elle veut utiliser. Ceci est fait dans un fichier XML du projet qui est présenté à l'utilisateur lors de l'installation. Dans le cas de l'application que nous vous présentons ci-dessus, android.permission.INTERNET est tout simplement la permission du logiciel à pouvoir communiquer sur Internet via les différents protocoles mis à disposition sur l'appareil (ex : Wifi, 3G, …).

4. Décomposition du projet

Nous avons deux parties distinctes dans le projet. La première englobe le serveur web et son interface :

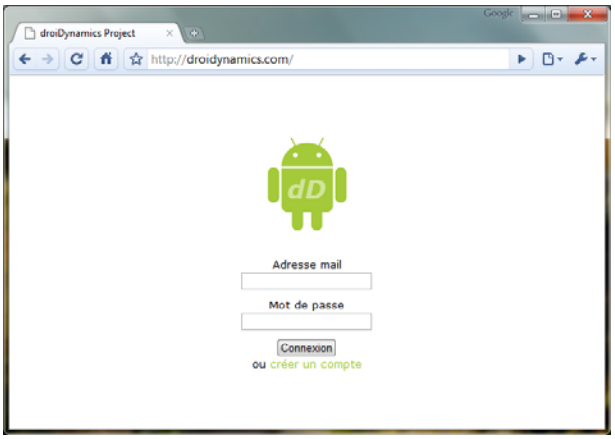


Figure 3 : Image de notre interface de login

Figure 2 : Image de notre interface d'ajout de point d'intérêt

et la deuxième englobe le programme sur la plateforme Android :

Figure 4 : Image de notre application tournant sur un émulateur du Smartphone HTC Magic

Plus précisément, l'interface web va s'occuper des points suivants :

- Création d'un compte
- Définition et suppression des points d'intérêt
- Définition et suppression de chaque ordinateur à démarrer par point d'intérêt

Et la partie sur le Smartphone, les points suivants :

- Géolocalisation de la personne via son téléphone portable
- Récupération des points d'intérêt
- Liste des points d'intérêts à proximité de la personne pour démarrer les ordinateurs
- Wake On Lan d'ordinateur

5. Technologies

Pour mieux comprendre la suite de ce rapport, voici quelques explications sur les technologies que nous avons intégrées dans notre projet. Tout d'abord celles en rapport avec la partie web :

5.1. Partie serveur sur Internet

5.1.1. SHA512

Le SHA est un algorithme de hachage qui génère une emprunte à partir d'une information donnée. Idéalement, un algorithme de hachage doit être protégé contre trois types d'attaques :

- Attaque sur la première préimage : Il doit être impossible de retrouver une information à partir de son emprunte.
- Attaque sur la seconde préimage : Il doit être impossible de trouver une autre information générant la même emprunte qu'une information et une signature données.
- Résistance aux collisions : Il doit être impossible de trouver deux informations aléatoires qui génèrent la même signature.

Les algorithmes de hachage sont fréquemment utilisés en informatique pour la protection des mots de passe (durant un échange et dans la base de données). Etant donné qu'il est impossible (en théorie) à partir d'une emprunte de retrouver l'information initiale (le mot de passe), la comparaison s'effectue entre les deux empruntes. Donc, quand un utilisateur entre son mot de passe dans un formulaire, il est haché et comparé à la signature qui avait été générée à partir de son mot de passe lors de la création de son compte.

Il existe différents algorithmes de hachage, en voici les principaux et les raisons qui nous ont poussées à en choisir un en particulier :

- MD5 (Message Digest 5) : C'est certainement l'une des fonctions de hachage les plus utilisées. L'emprunte est générée sur 128 bits. Cependant, il est déconseillé de l'utiliser pour des données critiques comme les mots de passe car une faille a été découverte depuis plusieurs années (résistance aux collisions inefficace).

- SHA-1 (Secure Hash Algorithm 1) : C'est également une des fonctions de hachage très utilisées sur Internet. On a tendance à déconseiller son utilisation étant donné que ces dernières années les attaques contre cet algorithme ont beaucoup évolué. Il est maintenant plus facile de calculer une collision que de faire une recherche exhaustive, donc cet algorithme n'est plus considéré comme fiable.

- SHA-2 (Secure Hash Algorithm 2) : Cet algorithme est tiré du SHA-1 en y apportant quelques modifications sur les calculs ainsi que sur la taille du résultat. SHA-2 peut produire des empruntes de 256, 384 ou 512 bits. Les dernières études en cryptanalyse ont démontré que les attaques employées sur MD5 et SHA-1 ne s'appliquent pas pour SHA-2, on peut donc le considérer comme sûr pour le moment.

Lorsque nous avons besoin de générer des empruntes, nous utilisons donc l'algorithme SHA-2 sur 512 bits. Nous avons choisi cet algorithme pour sa sécurité et parce que la taille de l'emprunte n'affecte pas de façon significative le temps de calcul comme le montre les calculs suivants :

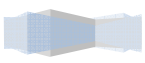
Fichiers utilisés :

```
17o 2010-01-06 09:38 0_petit
646M 2010-01-06 09:40 1_moyen
 4.4G 2010-01-06 09:48 2_grand

user@ks33104:~/temp$ time md5sum 0_petit
8db9abd55fa7b6cd9908627fb792f2bf  0_petit

real    0m0.002s
user    0m0.000s
sys     0m0.000s
user@ks33104:~/temp$ time sha1sum 0_petit
1c9b1f2baafa222718bf682ffbe6d3e7cebec5a0  0_petit

real    0m0.021s
user    0m0.004s
sys     0m0.000s
user@ks33104:~/temp$ time sha256sum 0_petit
81e904f6ba9027d4e3dc5fd19035c33269024f5ffb3c8fd787124882bdeafb6c  0_petit

real    0m0.017s
user    0m0.004s
sys     0m0.000s
user@ks33104:~/temp$ time sha512sum 0_petit
5e0f327eae5cd260c03b7cacbbeb185696799b0e5c6d074b5e436bdef2ce265268bc717c43c
70133a5bd7622db15b95561f208e3753792ab65fd8cb1b4b611ab  0_petit

real    0m0.023s
user    0m0.000s
sys     0m0.004s

user@ks33104:~/temp$ time md5sum 1_moyen
5b7886b359b65584fb0c2ce214650199  1_moyen

real    0m9.900s
user    0m2.752s
sys     0m0.820s
user@ks33104:~/temp$ time sha1sum 1_moyen
a87f010ebf2d0464c295d7300353296d106fb6c2  1_moyen
```

```
real    0m6.060s
user    0m5.332s
sys     0m0.452s
user@ks33104:~/temp$ time sha256sum 1_moyen
4995cfbeb5ebae706e4a350ce9b4c9b6c17fad5778149cdd5f9d6ed5333a5738  1_moyen

real    0m11.215s
user    0m10.669s
sys     0m0.300s
user@ks33104:~/temp$ time sha512sum 1_moyen
6625b40e7030a5d2b5d26df92fff587e82fba6066bc5e2c9da5bd1736ebf2ddd826c2c64a78
5736cf1fd2a0b392843b48579b32c2a44b8d8758458fab68cade5  1_moyen

real    0m7.557s
user    0m7.108s
sys     0m0.452s

user@ks33104:~/temp$ time md5sum 2_grand
7ea66fee666fa444f0256ab3c1ec6ba4  2_grand

real    1m38.767s
user    0m19.105s
sys     0m5.584s
user@ks33104:~/temp$ time sha1sum 2_grand
1e3e92b77eaeec17f5d77010f8b6a5df95181dc3  2_grand

real    1m10.548s
user    0m36.650s
sys     0m7.024s
user@ks33104:~/temp$ time sha256sum 2_grand
2926d69c8c4d4cdfda37701361c54cd1e46b83af98f0738699a9389b7234d20a  2_grand

real    1m50.680s
user    1m19.137s
sys     0m3.528s
user@ks33104:~/temp$ time sha512sum 2_grand
4cddf03d9be54240ef38fe62d3f05ca3387cee1350738edbac40c4fd040894b3ad71c963332
34b65a8504d93fd2983969c554b98a667b712cb07267c016fc8ec  2_grand

real    1m49.507s
user    1m1.860s
sys     0m3.936s
```

Résumé:

	17 octets	**646 Mo**	**4.4 Go**
MD5	0m0.002s	0m9.900s	1m38.767s
SHA-1	0m0.021s	0m6.060s	1m10.548s
SHA-2 (256)	0m0.017s	0m11.215s	1m50.680s
SHA-2 (512)	0m0.023s	0m7.557s	1m49.507s



Ces calculs ont été réalisés sur la configuration suivante :

- Système d'exploitation : Linux Debian 5.0.3
- Noyau : 2.6.27.10 64 bits
- CPU : Intel(R) Celeron(R) CPU 220 @ 1.20GHz
- Mémoire : 1 Go

5.1.2. HTML

Le HTML (HyperText Markup Language) est un langage balisé de mise en forme conçu pour les pages internet. Ce langage permet l'insertion de textes, d'images, de liens, de médias, de formulaires… On l'utilise fréquemment avec des feuilles de style CSS pour la mise en page et du JavaScript pour différents traitements du côté client. Les balises HTML se reconnaissent par leur syntaxe encadrée par des '<' et '>'. Par exemple, l'ouverture d'une balise titre se fera grâce au code suivant : '<title>'. La plupart des balises HTML ont une balise de fermeture avec la syntaxe suivante : '</balise>'. L'exemple complet du titre d'une page HTML serait : '<title>Mon titre</title>'.

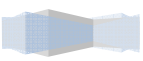
Analysons d'abord la structure d'une page HTML simple :

```
<html>
<head>
</head>
<body>
</body>
</html>
```

La balise <html> est obligatoire. Elle définit, avec son homologue </html>, respectivement le début et la fin du code HTML du document. La balise <head> permet d'intégrer au document certaines balises d'en-tête, tels que le titre du document (<title></title>) ou divers liens à des fichiers, CSS ou autres (<link>). Reste finalement la balise <body> qui encadre le corps (ou contenu « utile ») du document HTML.

Certaines balises HTML sont particulièrement intéressantes, on peut citer parmi celles-ci :

- '<form>' : Cette balise permet d'envoyer des données (fournies grâce à des champs '<input>') via un formulaire à une autre page sur le site internet, notamment une page exécutée sur le serveur afin d'y traiter les données saisies. Sa syntaxe est extrêmement simple : '<form action=''ACTION'' method=''METHOD''>…</form>'. La chaine qui remplace 'ACTION' permet de définir la page web qui recevra les valeurs du formulaire. Cette page peut-être identifiée soit par un lien relatif (form.php) soit par un lien absolu (http://monformulaire.com/form.php). La chaîne de caractères 'METHOD' permet de définir par quelle méthode seront envoyées (grâce au bouton '<input type=''submit''>') les valeurs du formulaire. Les deux valeurs admissibles sont :
  - GET : Les valeurs sont transmises dans la barre d'adresse, par exemple : http://monformulaire.com/form.php?champ1=valeur1&champ2=valeur2. Cette méthode n'est pas idéale pour les données sensibles car le visiteur de la page Web voit tout le contenu qui est envoyé et peut facilement le modifier.



- POST : Les valeurs du formulaires sont transmises dans le corps de la requête HTTP, par exemple :

```
GET /form.php HTTP/1.1
Host:monformulaire.com

champ1=valeur1&champ2=valeur2
```

Cette méthode est conseillée pour les formulaires sensibles malgré le fait qu'elle n'apporte pas beaucoup de sécurité supplémentaire. Les données sont toujours manipulables et visibles en interceptant les paquets transitant entre le client et le serveur Web, simplement ce n'est pas à la portée de tout le monde.

Ces différentes méthodes nous prouvent bien l'utilité de protéger autrement les données qui transitent entre nos formulaires et le serveur Web lorsqu'elles sont critiques. C'est pour cette raison que nous utilisons des astuces avec du chiffrement.

- Liens hypertextes : Les liens hypertextes représentent certainement une des parties les plus importantes du HTML. Ce sont eux qui permettent de passer d'une page à une autre à la demande du visiteur et de pointer sur des fichiers pour que l'utilisateur puisse les télécharger. Le succès et l'adoption du HTML pour Internet est certainement dû au concept de lien hypertexte. Le lien hypertexte a lui aussi une syntaxe extrêmement simple : 'TEXTE'. La destination est bien sûr remplacée par l'adresse (relative ou absolue) de la page que l'on souhaite atteindre en cliquant sur le lien et le texte permet d'insérer l'élément à cliquer. On pourrait aussi remplacer le texte par une image. Il est d'ailleurs possible de transmettre grâce à un lien des informations par la méthode GET, par exemple de façon simpliste : '<a href=''index.php ?utilisateur=visiteur''>'. Ceci nous permet d'aller à la page index.php en fixant la valeur 'utilisateur' à 'visiteur'.

- Images : Il est aisé d'ajouter des images à une page HTML. La balise à utiliser est la suivante : ''. En remplaçant ADRESSE_IMG par l'adresse relative ou absolue de l'image, elle s'affichera à l'endroit désiré dans la page HTML. A noter que puisque la balise '<img>' n'a pas de balise fermante, on la termine par '/>' au lieu du simple '>'.

C'est essentiellement ces balises HTML qui composent l'intégralité du site de droiDynamics (mise à part des balises de mise en page supplémentaires).

5.1.3. CSS

Pour modifier la présentation d'une page HTML, il suffit de créer une feuille de style CSS décrivant une balise HTML. Par exemple, pour mettre en Arial rouge chaque élément balisé ''<H1>'', il suffirait d'ajouter dans notre fichier CSS le code suivant :

```
H1 {
font-family : Arial ;
color :#FF0000 ;
}
```

La 1ère ligne de ce code nous indique que nous souhaitons modifier la présentation de toutes les balises H1 des documents liés à cette feuille de style. Avec la 2ème ligne, nous modifions la police du texte grâce au paramètre « font-family » fixé à Arial. La 3ème ligne fixe la couleur du texte avec le paramètre « color ». Nous fixons cette couleur à rouge, grâce au code hexadécimal RGB (Red : 255, Green : 0, Blue : 0).

La façon la plus simple d'utiliser une feuille de style est de l'intégrer directement au code HTML. Pour ce faire, il suffit de compléter le code HTML:

```
<html>
<head>
<style>
H1 {
font-family : Arial ;
color :#FF0000 ;
}
</style>
</head>
<body>
</body>
</html>
```

Il est également possible de définir une feuille de style externe à notre document HTML. Une fois de plus, il existe plusieurs façons de la lier à ce dernier :

1) En utiliser la balise <style> :

```
<html>
<head>
<style type="text/css" media="screen">
@import url(style.css);
</style>
</head>
<body>
</body>
</html>
```

Nous avons complété la balise <style> par les attributs vus précédemment dans la balise <link>. Ce code nous permet d'importer le fichier externe style.css afin d'y appliquer le code CSS qu'il contient.

2) En utilisant la balise <link>

```
<html>
<head>
<link href="style.css" media="screen" rel="stylesheet"
type="text/css" />
</head>
<body>
</body>
</html>
```

Les textes insérés entre les balises <H1> de chaque document HTML lié à notre CSS seront dorénavant écrits avec la police Arial rouge.

5.1.4. JavaScript

JavaScript est un langage de programmation essentiellement utilisé dans les pages Internet interactives. Il a été créé en 1995 par la société Netscape Communications Corporation mais c'est surtout ces dernières années qu'il a connu une importante utilisation avec le concept de web 2.0 (principalement caractérisé par Gmail, Facebook, Google Maps, …). En effet, ce n'est que ces dernières années que l'utilisation de AJAX (Asynchronous JavaScript and XML) a rendu extrêmement populaire JavaScript pour charger de nouvelles données dans une page sans la recharger.

On voit d'ailleurs apparaître dans le monde des navigateurs une forte concurrence dans les moteurs de rendus JavaScript, ce qui est parfaitement compréhensible car de nombreux utilisateurs choisiront le navigateur avec lequel les pages sembleront les plus fluides (étant donné la quantité de JavaScript dans les sites populaires, le site le plus fluide est le site traitant le plus rapidement le JavaScript).

Google Maps est justement un très bon exemple de l'utilisation de JavaScript. Pour charger une carte Google Maps dans une page Web, ce n'est que du code JavaScript. Les interactions avec cette carte sont également programmées en JavaScript.

Le langage JavaScript est évidemment traité côté client, puisque c'est le navigateur qui s'en charge. Cela signifie que le code source JavaScript est entièrement visible au visiteur du site et qu'il ne faut donc baser aucune sécurité sur d'éventuelles méthodes programmées de cette façon. Les codes JavaScript peuvent aisément être désactivés voir même modifiés.

5.1.5. PHP

PHP (PHP : Hypertext Preprocessor) est un langage de programmation libre utilisé le plus souvent pour créer des pages Internet dynamiques. Il peut aussi être employé pour des programmes en ligne de commande, mais ce n'est de loin pas son utilisation la plus fréquente.

Depuis la version 5 de PHP, le langage dispose d'un support complet de l'orienté objet ce qui en fait un langage très complet.

Lorsqu'il est utilisé dans le cas d'Internet, le code PHP est exécuté côté serveur. Cela signifie que seul le serveur verra le code PHP et l'interprètera. Le client, quand à lui, ne recevra que le résultat du traitement PHP, dans la plupart des cas du HTML. PHP peut également générer des fichiers images, des fichiers pdf ainsi que tous les types fichiers encodés en texte.

De plus, la bibliothèque PHP est très riche, ce qui en fait un langage très complet. Il peut, sans configuration particulière (mise à part l'installation des extensions PHP nécessaires sur le serveur), générer des images, des PDF, effectuer du chiffrement, lire et écrire des fichiers, créer des sockets, etc. PHP peut également se connecter à un grand type de bases de données.

Puisque le code PHP n'est pas visible par le client, des informations secrètes comme des mots de passe peuvent être stockées dans le code. L'exemple le plus fréquent est bien sûr le stockage du mot de passe de connexion à une base de données, qui est souvent écrit en clair dans le code PHP.

La syntaxe de PHP est souvent décrite comme un mélange de C et Perl. La partie Perl se repère immédiatement avec les variables qui sont préfixées par un '$'.

Le choix de PHP n'a pas été fait au hasard. Un facteur important étant le prix global de l'installation, nous avons choisi d'installer un serveur Linux Debian avec un serveur web Apache avec les extensions PHP. Aucun frais de licence ne s'applique dans ce cas. De plus, la richesse de PHP est très intéressante. On ne se retrouve que rarement bloqué par PHP à qui il manquerait une fonctionnalité. Encore une qualité essentielle de notre point de vue, c'est la qualité de la documentation. Le site php.net fournit une documentation de très bonne qualité, avec de nombreux exemples et des contributions d'utilisateurs. PHP est également très stable et évolue de façon rapide étant donné qu'il dispose d'une grande communauté.

De plus PHP possède des fonctions pour modéliser le concept de session. Les sessions PHP sont implémentées en conservant les informations dans un pseudo-cookie sur le serveur et en ne transmettant que l'identifiant de session à l'utilisateur. L'utilisateur stocke cet identifiant dans un cookie (ou le répète en paramètre GET si le navigateur refuse les cookies) et le transmet à chaque connexion au serveur afin qu'il puisse repérer les données de cet utilisateur en particulier. Cette méthode n'est cependant pas à l'abri de différentes attaques tels que le vol de l'identifiant de session. Cette attaque est cependant bloquée (sauf si un client NAT vole l'identifiant d'un utilisateur du même réseau) par notre code étant donné que nous ajoutons l'adresse IP de l'utilisateur dans la session. Il est important de noter que PHP est également capable de travailler directement avec les cookies, mais du point de vue de la sécurité, ce n'est pas un bon choix pour stocker des données sensibles.

5.1.6. MySQL

MySQL est le système de base de données le plus utilisé, en concurrence avec Microsoft SQL Server et Oracle. MySQL a été développé dans un souci de performance. Les avantages de MySQL sont bien sûr la performance, mais aussi la stabilité et la facilité.

MySQL supporte une grande quantité de moteurs de base de données. Parmi ceux-ci, on peut citer les plus importants qui sont :

- MyISAM : C'est le moteur de base de données le plus performant. C'est le moteur par défaut de MySQL.
- InnoDB : Ce moteur gère les transactions et les clés étrangères, ce qui améliore la fiabilité des données de la base grâce aux contrôles des clés et des relations. On peut noter une légère diminution des performances de InnoDB face à MyISAM.
- NDB : Ce moteur est utilisé pour stocker les données dans des clusters MySQL.

Nous avons choisi d'utiliser InnoDB comme moteur de base de données étant donné que la haute performance n'est pas une des priorités de notre projet et que nous préférons profiter des avantages offerts par les clés étrangères et les relations.

Il est intéressant de noter que MySQL AB a été racheté par Sun Microsystems en janvier 2008 alors que ce dernier a également été acquis par Oracle Corporation en 2009. Il apparait alors un conflit d'intérêt pour Oracle Corporation qui possède depuis longtemps un produit sous licence (Oracle) et désormais son plus grand concurrent gratuit (MySQL). Nombreux sont ceux qui se demandent si Oracle poursuivra le développement de MySQL ou si le projet ne sera pas «coulé » au profit du produit payant.

5.1.7. XML

XML (Extensible Markup Language) est un langage balisé similaire au HTML à la différence près qu'il n'y pas de balises choisies pour effectuer certaines fonctions. Le choix de la balise est libre au programmeur, d'où le terme ''Extensible'' dans le nom. XML sert essentiellement à stocker ou transférer des informations représentables en structure hiérarchique. Le but du XML est de faciliter l'interopérabilité entre des systèmes hétérogènes pour la transmission d'informations. On peut en effet constater que la plupart des langages de programmation modernes ont des fonctions de traitement des fichiers XML. Pour facilité cette interopérabilité, le XML est donc du texte clair qui est structuré et qui peut-être validé grâce à une DTD (voir chapitre suivant).

Un document HTML peut donc être considéré comme un fichier XML dont la balise racine serait '<html>'. Pour nos explications, reprenons un exemple plus simple, par exemple une liste de personnes :

```
<racine>
<liste>
<personne>
Carlo
</personne>
<personne>
Claudio
</personne>
</liste>
</racine>
```

Il est nécessaire de définir qu'il s'agit d'un document XML. Donc nous ajoutons tout en haut du document la déclaration suivante :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
```

Explications : Le premier et le dernier signe à l'intérieur des parenthèses pointues < et > est un point d'interrogation. Tout de suite après le point d'interrogation du début doit se trouver `xml` (en minuscules). Différentes mentions peuvent suivre sous forme d'attributs. La mention du numéro de version XML est obligatoire et indispensable même dans la variante simple. Le "numéro de version" se réfère ici à la version de la spécification du langage XML. À l'heure actuelle, c'est la version 1.0 de XML qui prévaut. Puis vient l'attribut encoding qui mentionne le jeu de caractères que nous utilisons pour les données de ce fichier. Dans notre cas nous utilisons l'ISO-8859-1 qui correspond au jeu de caractères ISO pour les langues de l'Europe de l'ouest afin d'autoriser les lettres accentuées. Finalement, avec l'attribut standalone, nous pouvons faire savoir à l'avance à l'analyseur syntaxique XML si le fichier actuel se réfère à une DTD externe ou non :

1) Avec `standalone="yes"` nous mentionnons que la DTD se trouve **dans le fichier actuel**.
2) Avec `standalone="no"` nous faisons savoir que la DTD se trouve **dans un fichier séparé**.

5.1.8. DTD

La DTD (Document Type Definition, Définition de Type de Document) permet de décrire un modèle XML. Elle n'est vraiment indispensable que si plusieurs programmes utilisent ou génèrent un code XML compatible entre eux (par exemple le format SVG). Elle peut être intégrée au document XML ou se trouver dans un fichier joint.

Si l'on souhaite intégrer une DTD directement dans le document, la 1ère ligne ne change pas, le document reste « standalone ». Il suffit d'intégrer juste après la 1ère ligne le code définissant notre DTD :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE racine [
<!ELEMENT liste (personne*)>
<!ELEMENT personne…
]>
<racine>
.
.
.
```

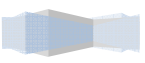
La ligne <!DOCTYPE liste [permet de préciser quel document XML elle spécifie en lui fournissant la balise racine (dans notre cas <racine>). Suite à cela, on ajoute la liste des éléments que le document contient.

Une autre solution pour définir une DTD est de créer un fichier *.dtd et de le lier au fichier XML. Pour ce faire, il faut modifier la 1ère ligne du code XML (le fichier n'est plus standalone) et ajouter une ligne qui lie le fichier XML à la DTD :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE racine SYSTEM "definition.dtd">
<racine>
```

Comme pour la déclaration de la DTD interne au document XML, il est nécessaire d'inscrire la balise racine (dans notre cas <liste>) après le mot clé DOCTYPE. Ensuite figure le mot clé SYSTEM, suivi du chemin du document contenant le code de la DTD. Le contenu du fichier de la DTD contient le même code que l'ancienne DTD (sans la 1ère et la dernière ligne qui sont la déclaration du DTD). On ajoute cependant une déclaration similaire à celle du document XML :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
```

5.1.9. Google Maps API

Google Maps est un service de cartographie gratuit réalisé par Google. Ce service permet de visualiser des images satellites de très grande altitude à des images relativement proches. De plus, il est également possible d'avoir une vue plus standard en affichant uniquement les routes et les rues.

Figure 5 : Vue Satellite de Yverdon-les-Bains suite à une recherche sur Google Maps[4]

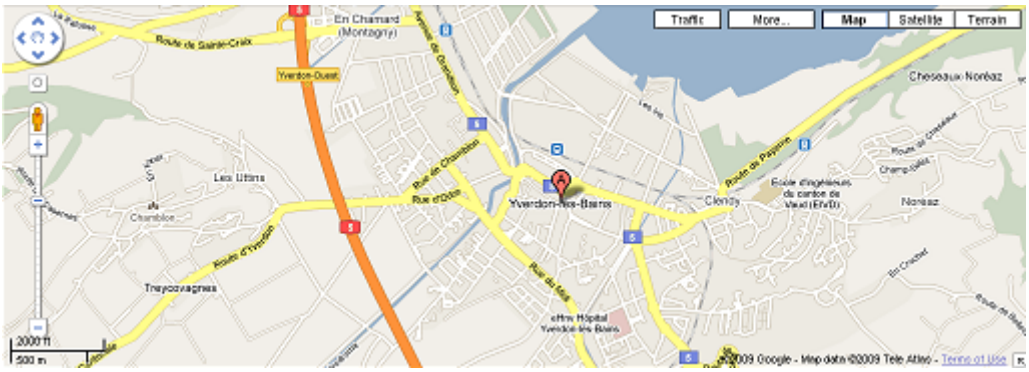


Figure 6 : Vue en mode carte de Yverdon-les-Bains suite à une recherche sur Google Maps[5]

Google fournit une API (Application Programming Interface) permettant d'intégrer et d'interagir avec le service Google Maps sur un site autre que celui de Google. Cette API est pour le moment entièrement gratuite et dépourvue de publicité, dans les termes du contrat d'utilisation, Google mentionne la possibilité d'ajouter des publicités au service.

Les possibilités d'interaction avec le service de Google Maps sont nombreuses. On peut sans problèmes ajouter des points à la carte, dessiner des formes, modifier le zoom, modifier l'apparence de la carte, gérer le drag&drop, etc.

Dans notre utilisation de l'API, il y a principalement deux éléments utilisés :

- Marker : Le marker est utilisé pour désigner un point sur la carte. Il est souvent représenté par le symbole officiel de Google :

  L'API de Google Maps nous permettant de modifier l'image d'un marker, nous avons préféré employer une image créée à cet effet et représentant mieux le projet :

[4] http://maps.google.com
[5] http://maps.google.com

Effectivement ce petit robot vert faisant un signe du bras représente la mascotte bien connue des utilisateurs de téléphones Android.

- Infowindow : Une carte Google Maps permet d'afficher une infowindow (une seule à la fois) qui se présente sous la forme d'une bulle de bande dessinée. Le contenu de cette bulle peut-être modifié à volonté en spécifiant simplement le code HTML qu'elle doit contenir. Grâce à ce code HTML il est possible d'y ajouter un simple texte, des images, des formulaires voir même des animations. Il est d'ailleurs possible de choisir l'événement déclencheur qui affichera la bulle (clic droit, double clic, …).

Figure 7 : Infowindow affichée sur la page d'accueil du site Web

5.2. Partie Android

Nous allons ici traiter les notions théoriques des technologies utilisées pour la partie du projet sur le Smartphone :

5.2.1. Android

Android[6] est une plateforme mobile libre, totalement personnalisable et open source. Elle contient un panel riche d'APIs qui permettent à des développeurs tiers de développer facilement de grandes applications.

Android est à la base développé par Goolge, mais la démarche de ce-dernier a été d'ouvrir rapidement son développement en rassemblant autour de lui et au travers de l'Open Handset Alliance un maximum de sociétés.

Cette volonté d'offrir un système de qualité et ouvert à tout le monde est venue du fait que notre façon de communiquer a beaucoup changé ces dernières années et continue à s'adapter aux changements importants liés à Internet. Nous sommes devenus beaucoup plus dépendants à Internet qu'auparavant grâce en grande partie aux maintes applications web de nouvelle génération qui remplacent petit à petit les logiciels installés sur nos postes de travail. Pour ne citer que quelques exemples les plus connus, prenons tout simplement les applications Google comme Gmail, Google Agenda ou encore Google Docs. Toutes ces applications ne sont plus installées sur nos postes de travail mais sont centralisées sur internet, c'est le « cloud computing ». Plus besoin de se soucier pour la sauvegarde de nos données ou de la réinstallation de nos logiciels en cas de formatage. Il

[6] Android doit son nom à la startup éponyme spécialisée dans le développement d'applications mobiles que Google a rachetée en août 2005, nom qui vient lui-même d'« androïde » qui désigne un robot construit à l'image d'un être humain. Source : http://fr.wikipedia.org/wiki/Android#Origine

nous suffit d'un simple accès à Internet et d'un navigateur pour tout avoir à portée de main. Le fait est qu'il nous faut tout, tout le temps et partout. C'est à cela que Google veut apporter sa solution! Grâce à un Smartphone petit et léger, nous avons accès à Internet via la 3G ou le Wifi pratiquement partout où nous nous trouvons. Ceci nous ouvre des perspectives intéressantes, telles que l'accès à notre moteur de recherche favori, nos applications, nos e-mails, nos horaires de bus, etc, partout et à tout moment.

Google profite là d'un énorme avantage car grâce à son savoir faire et à ses moyens, il a lancé un projet ambitieux qui va lui permettre de toucher encore plus de gens avec ses produits phares. N'importe qui pourra très facilement faire une requête sur Google où qu'il soit, se déplacer en utilisant Google Maps, lire et écrire ses emails avec Gmail, gérer son emploi du temps avec Google Agenda et plus encore.

Pour résumé, Android est un système d'exploitation destiné aux Smartphones, orienté Web et muni d'une multitude d'outils pour son utilisation et son développement.

Les avantages de ce système comparé aux autres acteurs du marché est qu'il est open source, complètement paramétrable et conçu pour profiter pleinement de la puissance qu'offre Internet. Ceci n'est pas le cas de ses concurrents directs tels que l'iPhone d'Apple, Symbian OS de Nokia, Windows Mobile de Microsoft ou encore BlackBerry.

Android est jeune mais grâce au nombre des partenaires qui sont impliqués dans son développement, il commence à gagner des parts de marché de plus en plus importantes. Les consommateurs sont de plus en plus intrigués et séduits par cette solution moins chère et ayant plus d'avantages que ses concurrents actuels. Nous pouvons voir sur les graphiques suivants que les consommateurs commencent à vouloir acheter de plus en plus de Smartphones tournant sur Android. La progression qu'Android connait ces derniers temps est énorme. Beaucoup de personnes s'y intéressent car beaucoup de constructeurs l'adoptent :

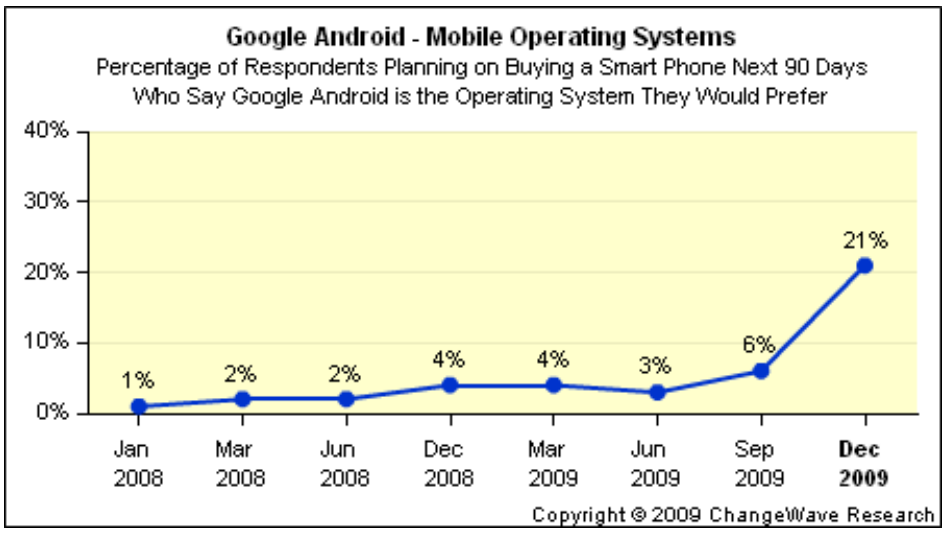


Figure 8 : Pourcentage de consommateurs qui planifient d'acheter un Smartphone Android dans les prochains 90 jours[7]

[7] http://www.changewave.com/assets/alliance/reports/smart_phone_20091221/google_android_future.gif

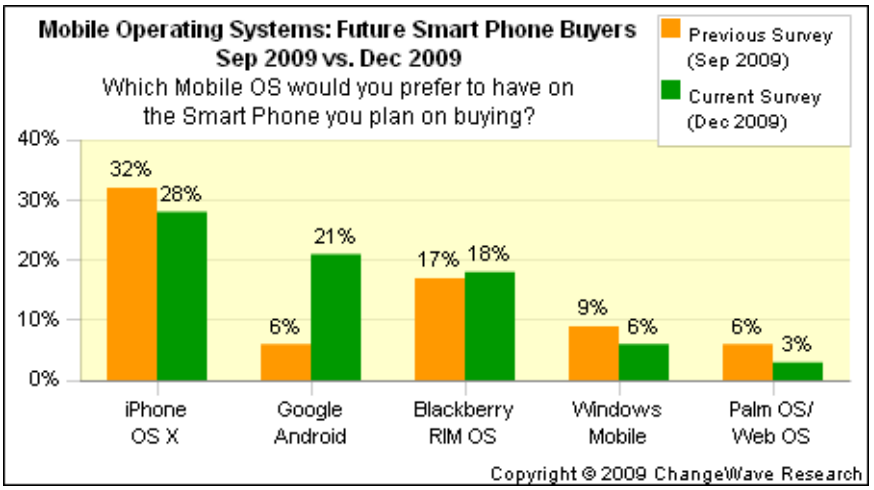


Figure 9 : Progression plus importante pour Android que pour ses concurrents[8]

5.2.2. Open Handset Alliance

L'Open Handset Alliance est un ensemble de sociétés qui œuvrent au développement d'Android. Parmi ces sociétés, nous y trouvons des fabricants de téléphones, des opérateurs de téléphonie, des sociétés liées à Internet, des constructeurs de puces électroniques ou encore des acteurs du marché des GPS.

Le nombre de partenaires ne cesse de s'agrandir. Vous pouvez trouver la liste de tous les partenaires sur le site de L'Open Handset Alliance à l'adresse suivante :
http://www.openhandsetalliance.com/oha_members.html.
Lors de l'écriture de ces lignes, il y a exactement 64 partenaires!

5.2.3. Licence

La license préférée d'Android est l' « Apache open source licence v2 » car contrairement aux licences GPL, elle autorise la reprise de tout ou partie du code dans un produit sous une autre licence y compris propriétaire. Le système d'exploitation lui-même est fourni avec la license Apache open source licence v2.

Sinon les licences autorisées sont celles open source approuvées par l'Open Source Initiative. La liste des licences autorisées est disponible sur le site : http://www.opensource.org/licenses/alphabetical.

5.2.4. Versions

Android a connu plusieurs versions jusqu'à présent :

- 1.1 - Bender
- 1.5 - Cupcake
- 1.6 - Donut
- 2.0 - Eclair
- 2.0.1 - Eclair –mise à jour
- 2.1 - Flan

Concernant le nom de code des versions, Il s'agit à chaque fois d'un nom de pâtisserie pris dans l'ordre alphabétique.

[8] http://www.changewave.com/assets/alliance/reports/smart_phone_20091221/mobile_os_future.gif

5.2.5. Architecture logicielle

Android se compose d'un système d'exploitation basé sur le noyau de Linux v.2.6 qui a la tâche de gérer le matériel à l'aide de drivers, la mémoire, les processus ou encore le réseau. Android se base sur l'ensemble Linux/Java et propose un Framework de développement en Java et en C.

Android propose aussi une suite de bibliothèques C/C++ très utiles et qui offrent des possibilités intéressantes. On retrouve libc, des librairies pour l'OpenGL ES qui tend à devenir la norme en matière d'API 3D pour les mobiles, le Webkit, SQLite, …

Webkit est un moteur web léger, puissant et qui respecte bien les standards en vigueur. SQLite est un moteur de base de données sous forme de librairie C.

De plus, Android met à disposition un middleware qui est l'environnement d'exécution. Ce middleware dispose d'une machine virtuelle Java modifiée appelée Dalvik. Modifiée car les besoins spécifiques des Smartphones imposait d'alléger la machine virtuelle standard.

Finalement, les applications sont codées et compilées avec le JDK 1.5 et 1.6 mais ne comprennent pas la totalité du JDK (plutôt un sous-ensemble optimisé pour Android).

Voici un graphique qui représente bien tous les éléments de l'architecture Android avec ses différentes couches :

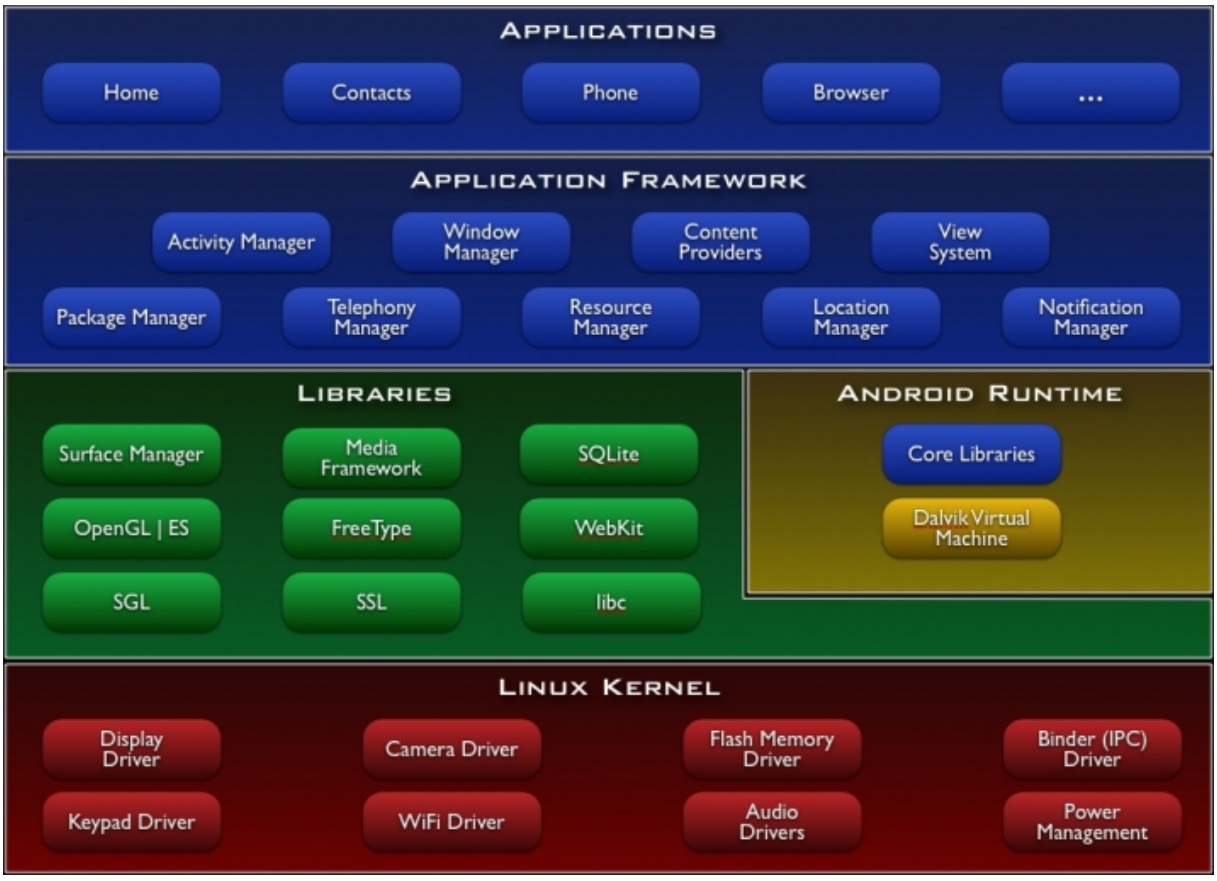


Figure 10 : Architecture Android[9]

[9] http://developer.android.com/images/system-architecture.jpg

Pour plus d'informations concernant l'architecture Android, le lien suivant vous sera utile : http://developer.android.com/guide/basics/what-is-android.html.

5.2.6. SDK

Le SDK fourni est un kit de développement ou trousse de développement logiciel. C'est un ensemble d'outils permettant aux développeurs de créer des applications en Java pour les diverses version d'Android.

Le SDK contient toute la documentation d'Android, des exemples de codes, des utilitaires à intégrer à notre IDE ou à utiliser en ligne de commande, des APIs Google tels que Google Maps, la bibliothèque Java (classes des packages propres à Android ainsi que les classes du JDK) et des drivers pour Windows afin d'installer les Smartphone Android (sur Linux ou Mac, aucun driver n'est à installer).

Ce SDK a été fortement développé pour être intégré à l'environnement de développement Eclipse. Il est très facile à installer et la plupart des tâches sont automatisées au sein de l'IDE. Grâce à la combinaison Eclipse/SDK Android, le développeur a des outils fonctionnels pour implémenter correctement une application et pouvoir la débuguer ainsi que l'émuler.

Grâce à ce SDK, nous retrouvons un environnement complet de débogage. Nous pouvons voir ci-dessous le debug de notre application avec l'émulateur :

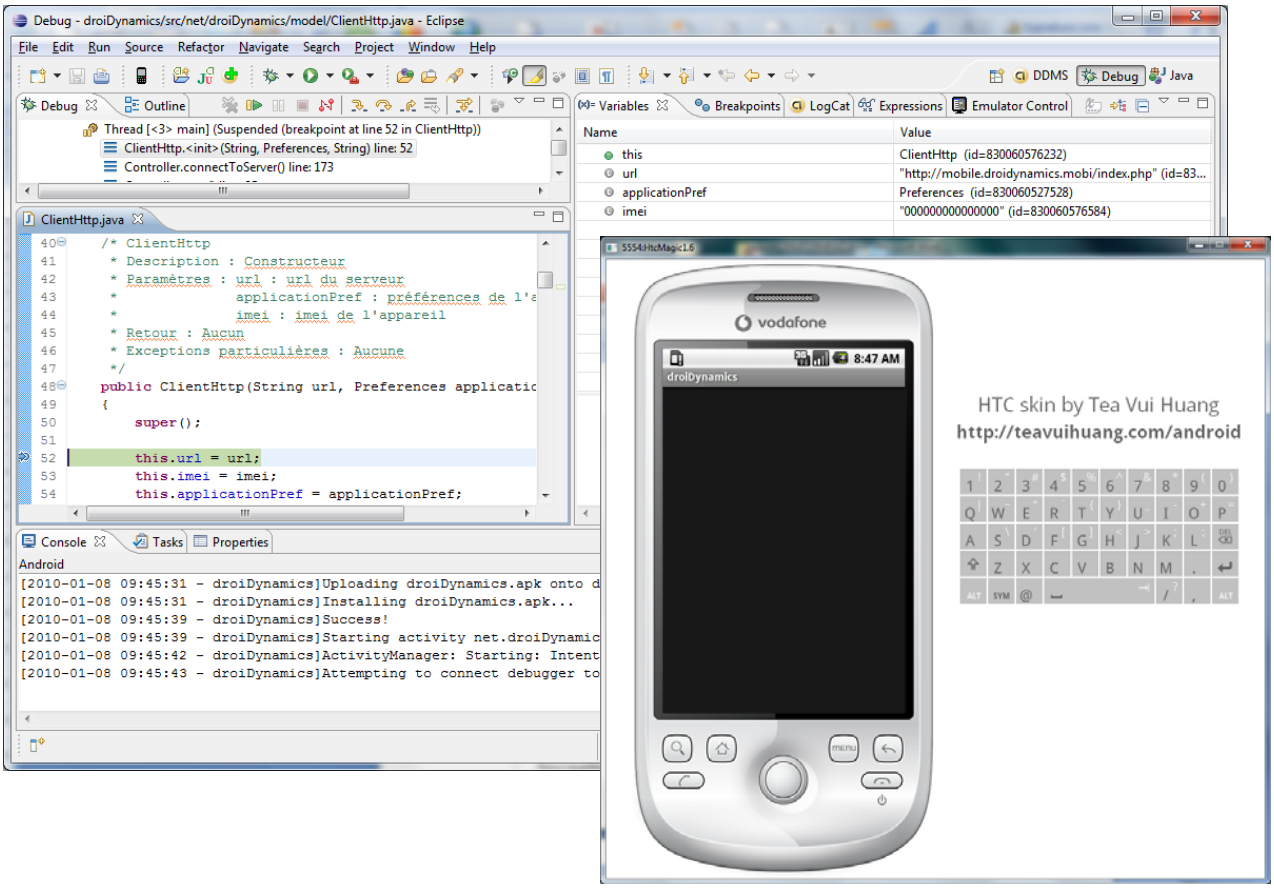


Figure 11 : Notre environnement de développement sur Eclipse Galileo avec le SDK d'Android et l'émulateur

5.2.6.1. Structure d'un projet

Tout d'abord lors de la création d'un projet Android, il faut remplir les informations suivantes :

- Nom du projet
- Emplacement du projet
- Pour quelle version d'Android cette version est destinée
- Nom de l'application
- Nom du paquetage
- Pour quelle version du SDK au minimum ce projet est destiné (facultatif)

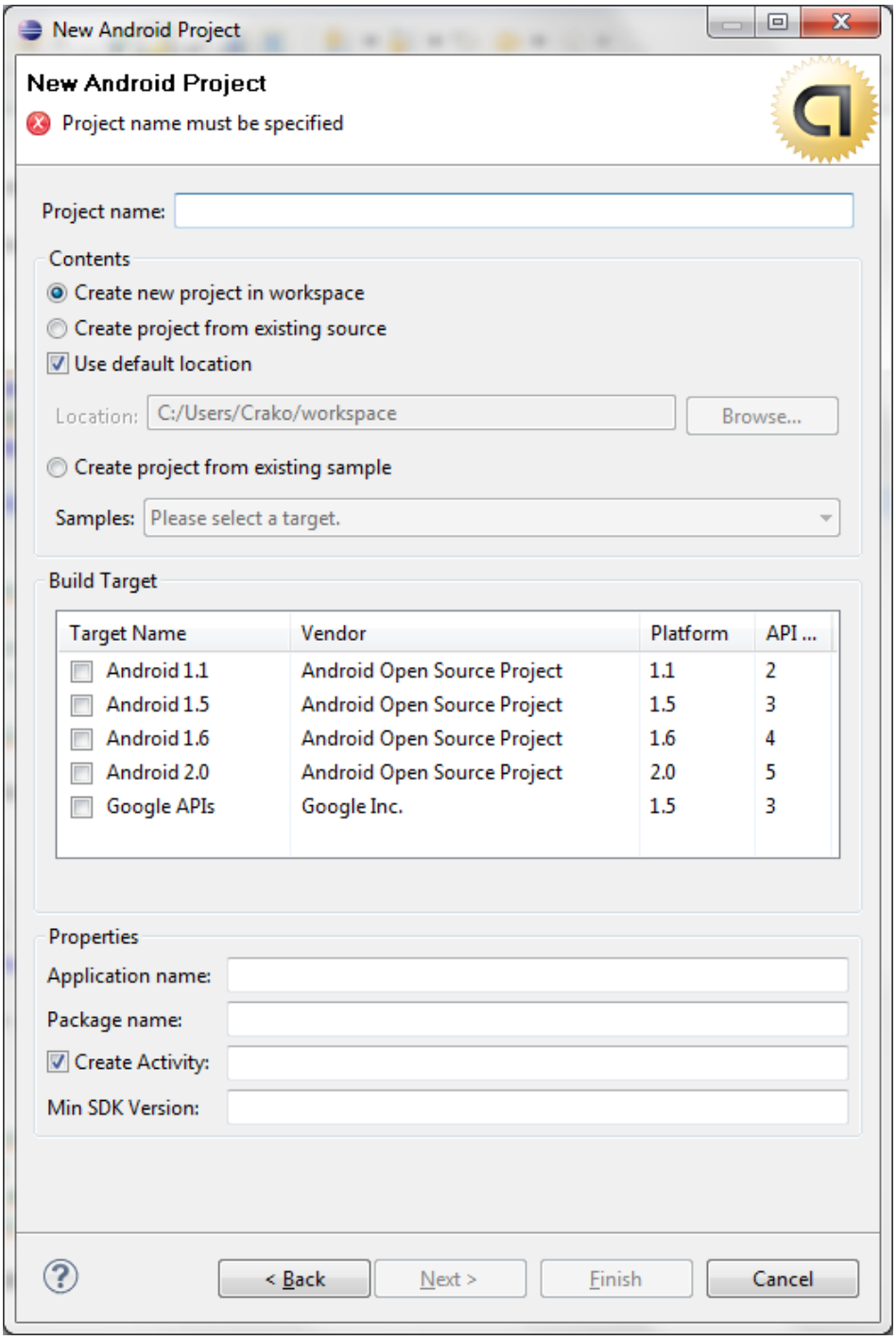


Figure 12 : Ajout d'un nouveau projet sur Eclipse utilisant le SDK Android

Puis notre projet se présente avec la structure suivante :

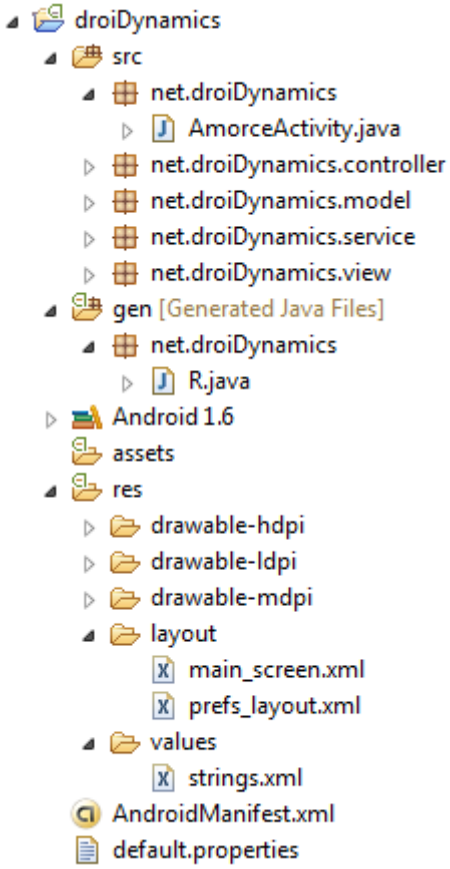


Figure 13 : Structure du projet

Notre projet contient 5 éléments importants et essentiels :

- Les sources dans le répertoire « src », qui se compose des paquetages et des différents fichiers .java
- La classe R qui est gérée automatiquement par le SDK et permet d'accéder à l'ensemble des ressources de notre application. Toutes les ressources (images, sons, texte) sont situées (avant packaging) dans le répertoire res mais ne devront être accédées qu'à l'aide de cette classe R.
- Toutes les classes Java Android dans le répertoire « gen » comprenant le fichier .jar se touvant dans le répertoire Android 1.6.
- Toutes nos ressources dans le répertoire « res ». Nous pouvons y retrouver nos images dans les répertoires « drawable-hdpi », « drawable-ldpi » et « drawable-mdpi » (répertoires différents en fonction des différents types de résolutions possibles des appareils cibles), nos « layouts » qui sont des fichiers XML représentant la structure visuelle d'une activité dans notre programme ainsi qu'un fichier XML contenant des ressources diverses (chaînes de caractères, tableaux, …) que nous pouvons utiliser dans nos layouts ou dans notre code.
- Finalement, le fichier AndroidManifest.xml qui contient des informations sur l'application et sur la librairie de notre projet. Diverses informations en métadonnées y sont regroupées.

Bien entendu, nous reviendrons sur toutes ces options et fichiers dans la partie concernant notre projet.

5.2.7. Composants d'une application

Il y a quatre composants possibles pour gérer et créer une application Android :

5.2.7.1. Activity

Le concept d'Activity repose sur la notion d'interaction utilisateur. Une Activity représente la fenêtre ou tout simplement l'écran qui sera affiché à l'utilisateur. Elle permet également de gérer des fonctionnalités telles que l'appui sur la touche [MENU] ou l'affichage de messages d'alerte.[10]

Pour mieux comprendre ce concept, nous pouvons apparenter tout simplement une Activity à une classe qui gère une interface utilisateur en lui affichant des layouts et en gérant les interactions

[10] http://www.pointgphone.com/tutoriel-composants-android-2238

possibles. Elle passe par plusieurs états pendant sa durée de vie sur le système d'exploitation Android. Voici un schéma qui représente les différents états :

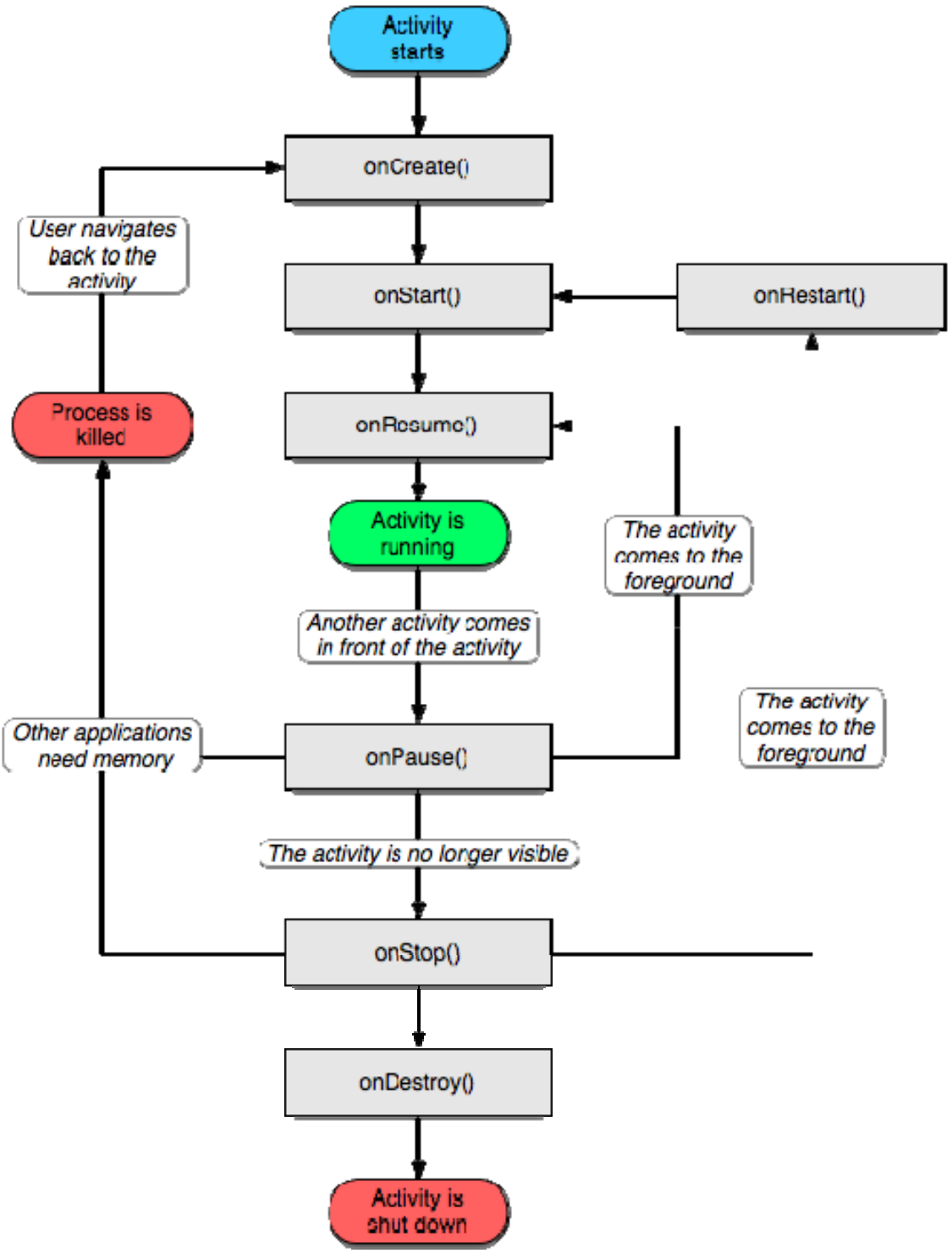


Figure 14 : Différents états d'une Activity[11]

[11] http://developer.android.com/images/activity_lifecycle.png

Nous pouvons voir qu'il y a quatre états importants :

- onCreate : Lorsque l'Activity est créée
- onStart : Lorsque l'Activity est démarrée
- onStop : Lorsque l'Activity est stoppée par le système par exemple quand une autre application est mise en premier plan ou quand une autre Activity de notre application est démarrée. Lorsque notre Activity revient au premier plan, elle va passer par l'état onStart.
- onDestroy : Lorsque l'Activity est détruite car elle a tout simplement fini son exécution

onResume, onRestart et onPause sont des états nécessaires à implémenter que dans peu de cas car une Activity qui n'a pas le premier plan sur le Smarphone, passe automatiquement par les états onStop et par la suite onStart.

Il faut savoir que le système d'exploitation peut à tout moment fermer notre application s'il a besoin de ressources (mémoire par exemple) pour l'exécution de tâches plus importantes. Ceci est tout simplement dû au fait que nos Smartphones sont limités dans la capacité et la performance de leurs composants électroniques.

De plus, nous pouvons définir l'Activity comme le maillon central d'une application. Par exemple au démarrage d'une application, le système crée et démarre tout simplement notre Activity principale.

5.2.7.2. Service

La notion de service est un avantage certain sur l'iPhone. En effet, un Service est en fait un programme tournant en tâche de fond et n'ayant pas d'interface graphique (ce qui n'est pas réalisable, à l'heure où j'écris ces lignes, sur iPhone OS). L'exemple commun illustrant au mieux cette notion est celui du lecteur mp3. Un lecteur mp3 ne nécessite pas, pour la plupart du temps, d'interface graphique et doit tourner en tâche de fond laissant la possibilité aux autres applications de travailler/s'exécuter librement.[12]

Comme cette notion de service nous est connue, nous pouvons vite comprendre l'intérêt d'un service sur un Smartphone. Grâce à lui, nous allons pouvoir exécuter des tâches de fond pour notre application même si l'utilisateur est en train de faire autre chose sur le téléphone. Par exemple, dans notre application, nous allons faire tourner un service qui va gérer les mises à jour de la position de l'utilisateur afin de lui proposer les bonnes fonctionnalités quand il les demandera. Bien entendu cette tâche ne doit pas être « bloquante » pour l'utilisateur et surtout elle ne doit pas gêner le fonctionnement des autres applications.

Le système va donc laisser tourner les services qui sont créés même si l'Activity qui les a créés n'est plus visible et active. Pour cela, le service va disposer de plusieurs états lors de son exécution. Ce graphique représente la suite logique des états possibles lors de l'exécution d'un service :

[12] http://www.pointgphone.com/tutoriel-composants-android-2238

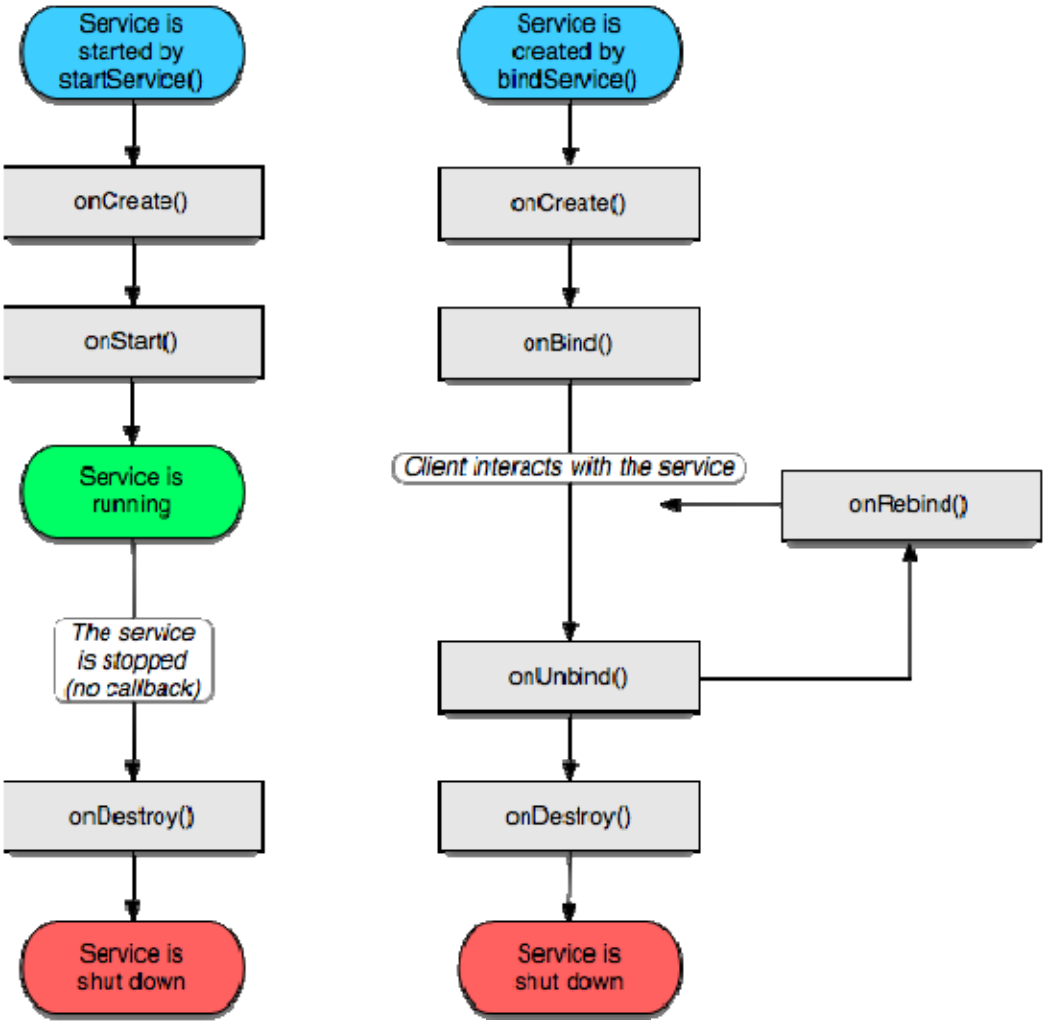


Figure 15 : Différentes étapes dans la vie d'un service[13]

Nous pouvons constater tout d'abord qu'il y a deux façons d'appeler un service :

- L'Activity crée son service avec startService() mais n'interagit pas avec les fonctions qu'il met à disposition
- Une Activity peut interagir avec un service en utilisant son interface via bindService() qui va lui retourner ce qu'il faut pour que l'Activity puisse appeler les fonctions mises à disposition par le service

Dans notre cas, nous allons utiliser la première méthode car nous allons lancer un service qui va se gérer tout seul et interagir tout seul avec l'utilisateur.

Nous pouvons constater que à la différence d'une Activity, le service n'a que les états onCreate, on Start et onDestroy. Ceci est dû au fait que le service n'a pas à être mis en attente car il ne possède pas d'interface visuelle (Layout). Il est destiné à faire des tâches de fond sans être dérangé.

[13] http://developer.android.com/images/service_lifecycle.png

Malheureusement, comme nos ressources sont limitées sur le Smartphone, nous avons découvert lors de nos tests que le système d'exploitation peut détruire un service pendant un petit moment pour libérer des ressources et le redémarrer automatiquement par la suite. Cette situation est très rare, nous l'avons rencontrée en moyenne qu'une fois par 24 heures d'utilisation et lors d'une utilisation très intensive du Smartphone. De plus le temps d'inactivité du service est extrêmement courte, dans nos tests elle ne dure même pas une seconde (ce temps peut très certainement varier en fonctions des besoins en ressources de l'OS au moment où il décide d'arrêter le service). Cet arrêt non programmé peut tout simplement s'expliquer par le fait qu'il existe des programmes et services plus importants que le notre, par exemple ceux qui concernent la gestion des appels ou du réseau.

5.2.7.3. ContentProvider

Les ContentProvider sont, comme l'exprime leurs noms, des gestionnaires de données. Ils permettent de partager l'information entre applications. Imaginons une application qui permette de conserver les cartes de visite virtuelles d'un ensemble de personnes. Ces cartes de visite contiennent généralement le nom, le prénom et un moyen de contact de la personne. Un tel programme peut être créé sous forme de ContentProvider ce qui lui permettra de fournir à d'autres applications présentes sur le système les informations sur une personne. Une application tierce d'envoi de courriel peut par exemple interroger ce ContentProvider afin d'obtenir l'adresse courriel d'un contact.[14]

Dans notre programme nous n'utilisons pas de ContentProvider.

5.2.7.4. BroadcastReceiver

Un BroadcastReceiver est un composant d'une application qui est à l'écoute de messages sur le système. Nous pouvons apparenter le concept de BroadcastReceiver par un bus de messages. Tous les messages circulent dans un bus avec un certain identificateur. Notre BroadcastReceiver va donc s'abonner à notre bus en lui passant comme paramètre à quel identificateur de messages il veut s'abonner.

L'identificateur est en réalité un Intent qui vous sera expliqué dans la suite du rapport. Pour résumer le processus, des programmes peuvent faire un broadcast d'un message avec un certain Intent et peuvent s'abonner pour recevoir tous les messages comportant un certain Intent.

Ceci est la manière dont les Activity, les Services et les ContentProviders communiquent entre eux.

5.2.8. Intent

Dans le système Android, un Intent est une « intention ». Un Intent est composé de trois attributs essentiels qui participent à l'identification :

- Action : chaîne de caractères qui symbolise le traitement à déclencher
- Data/Type : donnée qui détaille l'action
- Category : classification de l'action

De plus un Intent possède encore d'autres attributs tels que les « flags » ou les « extras » qui sont utilisés pour ajouter des informations pour le réceptionneur de l'Intent.

L'Intent est utilisé dans le cadre des BroadcastReceivers comme expliqué dans le point précédent, mais son utilisation principale réside dans l'appel d'Activity, de Services ou de ContentProviders. Pour

[14] http://www.pointgphone.com/tutoriel-composants-android-2238

comprendre ce concept, il faut tout d'abord savoir qu'une Activity ou un Service, peuvent démarrer d'autres Activity ou Services. De plus, chaque Activity ou Service est identifié avec une ou plusieurs actions (Exemple : composer un numéro, faire un calcul, …). Donc finalement, nous pouvons comprendre que pour lancer une nouvelle Activity depuis une Activity principale, il faudra préciser quelle Activity avec quel Intent (intension d'action) on veut démarrer.

Nous utilisons à divers endroits de notre code des Intent. Lors de leur utilisation, nous expliquerons plus en détails leur fonctionnement.

5.2.9. Appareils présents sur le marché

Il y a 19 terminaux actuellement disponibles sur le marché : Acer Liquid, Android Dev Phone 1, Dell Streak, GeeksPhone One, HTC Bravo, HTC Dream, HTC Eris, HTC Hero, HTC Magic, HTC Tattoo, Huawei U8230, LG GW620, Motorola Dext, Motorola Milestone, Samsung Behold II, Samsung Galaxy, Samsung Moment, Samsung Spica et Sony Ericsson Xperia X10.

Nous pouvons donc constater que même si Android n'en est qu'à ses débuts, l'offre est assez variée et complète. Bien entendu, ces appareil diffèrent par leurs composants internes et donc par les fonctionnalités qu'ils offrent aux utilisateurs.

Pour notre projet, nous avons travaillé sur un HTC Magic avec Android 1.6, mais il faut savoir que toute application développée pour Android peut être exécutée sur n'importe quel teminal ayant au minimum la version de l'OS pour laquelle l'application a été développée (les versions supérieurs sont rétro compatibles mais exigent dans peu de cas quelques adaptations du code de l'application).

Lors de l'écriture de ces lignes, une énorme annonce de Google fait l'actualité : la sortie de son Smartphone Nexus One. Nous somme tout particulièrement intéressés et nous nous réjouissons tout particulièrement de cette nouvelle car ce téléphone est tout simplement le Smartphone le plus complet et le plus puissant du marché. Il fonctionne sur une nouvelle version d'Android, la version 2.1 et possèdes de nouvelles applications telles qu'un navigateur GPS fonctionnant grâce à la richesse de Google Maps. Voici ses caractéristiques, ce qui vous donnera un aperçu de ce qu'un terminal sur Android peut offrir à ces utilisateurs :

Le Nexus One dispose d'un écran AMOLED 3,7 pouces, avec une résolution 800 x 480 pixels, un contraste de ratio 100 000:1 et un temps de réponse de 1 ms. Il possède deux micros, l'un à l'avant et l'autre à l'arrière, afin de permettre la réduction du bruit ambiant lors de conversations téléphoniques.

Le téléphone est équipé d'un processeur Qualcomm QSD 8250 Snapdragon cadencé à 1 Ghz, de 512 Mo de RAM, 512 Mo de mémoire flash interne et d'une carte microSD de 4 Go. Son appareil photo de 5 mégapixels avec autofocus et zoom numérique 2x est accompagné d'un flash LED. Il dispose également d'un récepteur A-GPS, du Bluetooth 2.1 + EDR, et du Wi-Fi 802.11b/g/n. Il est capable de lire les formats vidéos H.263, H.264 et MPEG-4, les formats audios MP3, AAC+, Ogg Vorbis, WAV, et MIDI, et d'afficher les formats d'images JPEG, GIF, PNG, et BMP. Il possède un port Micro USB standard à la place du connecteur HTC propriétaire, une prise casque 3,5 mm, un haut-parleur et un emplacement pour carte microSD permettant d'étendre la mémoire jusqu'à 32 Go. Les applications ne peuvent être installées que sur les 512 Mo de mémoire flash interne.

Le téléphone pèse 130 grammes avec la batterie, a une longueur de 119 millimètres, une largeur de 59,8 millimètres, et une épaisseur de 11,5 millimètres. Il est alimenté par une batterie amovible de 1400 mAh, qui lui permettrait de tenir jusqu'à 290 heures en veille (2G), 10 heures en conversation (2G), ou 5 heures en navigation sur internet (3G).[15]

Figure 16 : Nexus One de Google construit par le fabriquant de Smartphones HTC[16]

Nous pouvons donc constater que cet appareil fourni une multitude d'outils pour les programmeurs! Nous pouvons citer en plus l'intégration de la reconnaissance vocale dans Android 2.1 qui pourra ouvrir encore plus de perspectives pour les applications à venir.

[15] http://fr.wikipedia.org/wiki/Nexus_One
[16] http://www.futura-sciences.com/uploads/tx_oxcsfutura/NexusOne_Google.jpg

6. Principe général de fonctionnement

6.1. Serveur Web

6.1.1. Fonctionnement du site internet

6.1.1.1. Site complet

Le site complet permet à l'utilisateur de créer un compte, de gérer ses zones grâce à l'interface Google Maps et de gérer ses ordinateurs à réveiller. Ce site est accessible grâce aux adresses suivantes :

- http://www.droidynamics.com
- http://www.droidynamics.org
- http://www.droidynamics.net
- http://www.droidynamics.ch
- http://www.droidynamics.mobi

La 1ère page du site est simplement une page d'identification. On y retrouve les champs d'adresse mail et de mot de passe ainsi qu'un lien permettant de créer un compte.

Figure 17 : Ecran d'accueil du site Web

La création du compte se réalise grâce à une page permettant d'inscrire les informations essentielles d'un utilisateur. Nous avons préféré ne pas poser une grande quantité de questions à l'utilisateur à la création du compte, informations qu'il peut sans problèmes fournir plus tard.

Figure 18 : Ecran de création d'un compte sur le site Web

L'utilisateur peut alors compléter les mêmes informations qui apparaissent à l'identification. Une des différences est que le mot de passe demande une confirmation, afin d'éviter qu'une faute de frappe n'empêche l'accès au site et qu'il demande la création d'un nouveau compte. Le site demande aussi, dès l'enregistrement, un numéro IMEI (qu'il est possible d'obtenir grâce au code *#06# sur la plupart des téléphones mobiles) et une description de ce numéro. Etant donné que les codes IMEI sont enregistrés de façon hachée dans la base de données, une description est indispensable afin de pouvoir différencier un code d'un autre.

Une fois identifié, la navigation sur le site Internet se fait grâce au menu dans la partie haute de la page. Ce menu permet d'accéder à l'accueil (contenant l'interface Google Maps), à la configuration des accès et au bouton pour se déconnecter.

Accueil - Changer les paramètres d'accès - Se déconnecter

La page d'accueil présente sur la partie gauche la carte avec les zones existantes. Le fait de cliquer sur le petit robot vous indiquant une zone vous redirigera sur la gestion de la zone. Un clic avec le bouton droit sur la carte vous permettra d'ajouter une zone à votre compte grâce à une InfoWindow dans laquelle il faudra remplir quelques informations essentielles :

Figure 19 : Infowindow d'ajout de zone

Les coordonnées de la zone sont indiquées automatiquement par le système en fonction du lieu où vous avez cliqué. Il ne reste plus qu'à indiquer le nom que vous souhaitez lui donner et l'adresse IP de cette dernière (ou un nom de domaine). Après la validation du formulaire, la page est rafraichie et un nouveau robot apparait pour indiquer une nouvelle zone.

Une liste à droite de la page vous permet également d'avoir une vue rapide de toutes les zones qui sont enregistrées. Un clic sur une zone de la liste vous redirigera aussi sur la page d'édition de la liste.

Figure 20 : Page de modification d'une zone

Il est alors possible de modifier les paramètres de la zone (qui ont été saisis lors de sa création) ou d'ajouter des ordinateurs à réveiller. On peut également noter la présence d'un lien « supprimer la zone » sous le bouton de modification.

Comme présenté auparavant, pour chaque ordinateur à réveiller, il est nécessaire de fournir son adresse MAC ainsi que le port NAT qui permettra de rentrer dans le réseau (3G) ou le port 9 si le réveil se fait par WIFI.

Figure 21 : Liste des ordinateurs de la zone

Pour la suppression d'un ordinateur, un lien « supprimer » prend place où se trouvait le bouton d'ajout de l'ordinateur. Un simple clic sur ce bouton vous permettra de ne plus gérer cet ordinateur.

Une autre partie du site permet de gérer les accès à droiDynamics, autant sur le site Web complet que par le téléphone mobile.

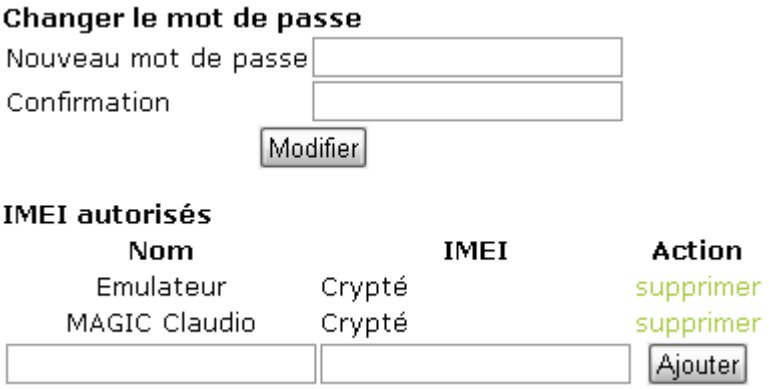


Figure 22 : Page de modification des paramètres d'accès

Un premier formulaire nous permet de modifier le mot de passe d'accès au compte (sur le site complet et via le programme mobile). L'ancien mot de passe n'est bien sûr pas affiché étant donné qu'il est enregistré sous forme d'emprunte dans la base de données.

La 2ème partie de la page nous permet de gérer les numéros IMEI qui sont autorisés à contacter le site via l'interface mobile. Cette partie ne concerne donc aucunement l'accès au site internet pour l'administration. On peut constater que les numéros IMEI ne sont, une fois de plus, pas affichés, étant donné qu'ils sont enregistrés sous forme d'emprunte, d'où l'importance d'avoir attribué une bonne description du code IMEI.

Une fois toutes les tâches d'administration terminées, il est important de se déconnecter en utilisant le lien approprié dans le menu afin d'éviter qu'une personne malintentionnée accède au site en utilisant votre ordinateur et la session sur le site non terminée.

6.1.1.2. Site mobile

Le site mobile n'est utilisé que par le programme Android pour se connecter et récupérer les fichiers XML. Il est accessible aux adresses suivantes :

- http://mobile.droidynamics.com
- http://mobile.droidynamics.org
- http://mobile.droidynamics.net
- http://mobile.droidynamics.ch
- http://mobile.droidynamics.mobi

Ce site est très simple dans la mesure où on s'y connecte, il nous retourne un jeton (chaîne aléatoire servant de sel pour l'authentification, voir chapitre suivant) et suite à cela, il suffit de se connecter à une page d'authentification pour recevoir la liste des zones :

1) Connexion à http://mobile.droidynamics.com
2) Récupération du jeton (chaîne aléatoire)
3) Connexion à http://mobile.droidynamics.com/?p=MOT_DE_PASSE_HACHE&m=ADRESSE_MAIL&i=IMEI_HACHE
4) Récupération des zones (ou d'une erreur)

La communication avec le site mobile est ainsi terminée et l'application peut effectuer le traitement qu'elle souhaite aux zones de l'utilisateur.

Pour le site mobile, souvent accédé par des réseaux lents de type EDGE ou 3G, nous avons essayé de mêler légèreté, simplicité (au niveau de la quantité de données) et sécurité.

6.1.2. Méthode d'authentification

L'idée de transmettre les informations sensibles d'authentification de façon sécurisée a représenté une priorité dans notre projet. Il était clair que nous n'allions pas transmettre ces informations en clair. En fait, la plupart des sites internet (mis à part les sites HTTPS) ne prennent pas en compte les dangers d'une attaque de type « Man in the middle » pour les authentifications. Nous avions alors deux possibilités, soit implémenter une infrastructure HTTPS, soit utiliser une astuce de programmation. N'ayant pas le temps de nous pencher sur HTTPS, nous avons préféré la 2ème solution, bien qu'elle ne soit pas autant fiable que HTTPS.

Nous avons d'abord pensé à simplement hacher le mot de passe de l'utilisateur avant l'envoi. Le problème de cette méthode est que, bien que le mot de passe en clair soit protégé, un pirate capturant la transmission peut se contenter de renvoyer l'emprunte qu'il a capturée pour s'authentifier. Nous avons alors décidé de mettre au point une autre solution.

Pour ce faire, nous avons élaboré le protocole suivant :

- Le serveur crée une chaine aléatoire (token) qu'il va hacher avec le même algorithme que le mot de passe et le transmettre au client
- Le client, avant de retransmettre une réponse au serveur, va hacher le mot de passe de l'utilisateur. Cette emprunte sera de la même longueur que le token reçu du serveur.
- Le client va mélanger un caractère du token avec un caractère du mot de passe et hacher le tout.
- Il va finalement transmettre cette emprunte au serveur.

De cette façon, le mot de passe n'est pas transmis en clair et le message transmis change à chaque authentification. Cette méthode repose cependant entièrement sur la complexité du résultat fourni par l'algorithme de hachage. Nous considérons que le hachage d'une valeur aléatoire nous fournit une emprunte suffisamment aléatoire et que l'algorithme (SHA512) en lui-même est sûr.

Voici un schéma résumant notre méthode d'authentification :

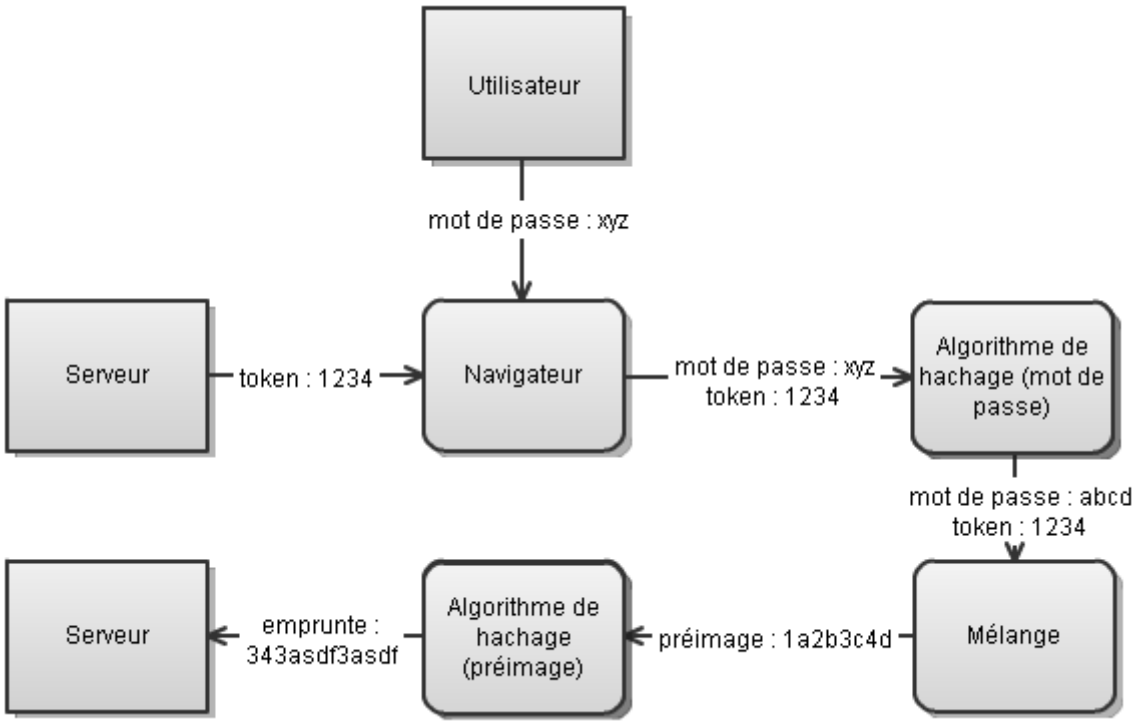


Figure 23 : Procédure de chiffrement d'une donnée sensible

Ce schéma ne couvre bien sûr que l'envoi d'une information sensible telle qu'un mot de passe. Il y a bien d'autres informations qui sont transmises.

De plus, nous avons choisi d'utiliser, comme précisé plus haut, l'algorithme de hachage SHA512.

6.1.2.1. Navigateur => Serveur

Pour que l'utilisateur s'identifie sur le site web de droiDynamics, l'authentification est très similaire à la méthode générale. Lorsque l'utilisateur arrive sur l'écran de connexion, il est déjà en possession d'un token caché dans le formulaire :

```
<input type="hidden" name="input_server_data" id="input_server_data"
value="c11486933d30e964d7010047f265cdc18648176e2f441bfa8f194f6f331604d37673
34f0e6747ac6bfd999ddecc3068cc376afb9e9e6be44d8a399da127b1ebc" />
```

Le navigateur, à l'envoi du formulaire, va donc hacher le mot de passe, le mélanger avec le token et hacher le tout pour l'envoyer au serveur avec l'adresse mail pour l'identification de l'utilisateur. Les deux informations nécessaires pour s'identifier via le site Internet sont l'adresse mail (identification) et le mot de passe (authentification).

Figure 24 : Ecran d'accueil du site Web

6.1.2.2. *Android => Serveur*

Le téléphone mobile Android enregistre les paramètres tels que le mail et le mot de passe de l'utilisateur. Cela signifie qu'en cas de vol du téléphone portable, le malfaiteur peut se connecter sans problèmes pour récupérer les zones et les points d'accès.

Une des solutions pour contrer ce problème aurait été de demander souvent les informations d'identification à l'utilisateur, mais le confort de l'application aurait été fortement réduit. En effet, un grand nombre de téléphones Android disposent d'un clavier virtuel ce qui n'est de loin pas idéal pour la saisie de longs mots de passe.

Nous avons préféré opter pour une solution plus confortable similaire au filtrage MAC des routeurs Wifi. Au lieu de nous baser sur l'adresse MAC d'une éventuelle carte réseau du téléphone, nous récupérons son numéro IMEI (International Mobile Equipment Identity) qui est un numéro unique attribué à chaque appareil GSM. L'utilisateur doit donc saisir ce numéro sur l'interface Web afin d'autoriser le téléphone à se connecter à l'interface mobile pour récupérer les données. Si le téléphone mobile est perdu ou volé, il suffit à l'utilisateur d'aller sur le site Internet de droiDynamics et du supprimer le numéro IMEI de son mobile de la liste des numéros autorisés.

De ce fait, nous avons trois informations nécessaires pour la connexion depuis le téléphone mobile (adresse mail, mot de passe et imei). L'adresse mail est transmise en clair (comme pour l'identification sur le site Web normal) et le mot de passe suit le même protocole de chiffrement (hachage du mot de passe, réception du token, mélange avec le token et hachage du tout). Par mesure de sécurité, nous avons préféré chiffrer l'IMEI aussi, avec le même protocole que le mot de passe. Pour éviter de demander trop d'informations au serveur, nous réutilisons le token du mot de passe durant le chiffrement de l'IMEI. Toute cette méthode est expliquée plus en détails dans la partie Android.

6.1.3. Echanges d'information web => Android

Nous avons vu précédemment que le serveur Web transmet des informations au client mobile. Pour tous les échanges, nous avons opté pour du XML, afin de garantir une grande compatibilité avec de

nombreux langages de programmation tout en gardant une certaine structure dans nos messages. Nous avons considéré les trois types d'informations suivants :

6.1.3.1. XML token

Le token est le code aléatoire généré par le serveur pour servir de sel lors du chiffrement des informations d'authentification. Le XML du token a une structure très simple :

```
<?xml version="1.0" encoding="utf-8" ?>
<token>49e74449dd5f75c2875c885e43b82a52ee4461e1bc203318830fc27d81d8ba27681f
4d2f4bb95e70a970cf86adf1e9c8c0929fffea20122758bce3c314391275</token>
```

On peut donc constater la présence du prototype XML suivi de la balise racine '<token>' qui englobe simplement la chaîne aléatoire.

6.1.3.2. XML erreurs

Un autre type de message généré est dû aux erreurs d'authentification. Etant donné qu'il peut y avoir plusieurs sources d'erreur, il y a également plusieurs messages possibles. En voici la liste :

- Structure : Il manque une information dans la requête d'identification. Par exemple, il manque le mot de passe, le mail ou l'IMEI. Ce type d'erreur génère le XML suivant :
  <?xml version="1.0" encoding="utf-8" ?>
  <error>structure</error>
- Mail : L'adresse mail fournie est invalide. Soit sa forme ne respecte pas le format des adresses mail (par exemple, il n'y pas de '@' ou de '.' ou l'extension fais moins de deux caractères ou plus de quatre), soit l'adresse mail n'existe pas dans la base de données. Cette erreur génère le XML suivant :
  <?xml version="1.0" encoding="utf-8" ?>
  <error>mail</error>
- Mot de passe : Cette erreur survient si le mot de passe n'est pas le même que le mot de passe associé à l'adresse mail. Le XML suivant est généré :
  <?xml version="1.0" encoding="utf-8" ?>
  <error>password</error>
- IMEI : Cette erreur survient lorsque l'IMEI fourni lors de l'identification ne fait pas partie des IMEI autorisés sur le compte. Voici le XML généré :
  <?xml version="1.0" encoding="utf-8" ?>
  <error>imei</error>

Il suffit au client mobile de gérer ces quatre types d'erreur afin de pouvoir avertir l'utilisateur du problème survenu de façon précise afin qu'il sache immédiatement quelle partie corriger.

6.1.3.3. XML zones

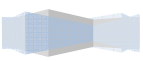
Le dernier type d'information que le serveur Web peut communiquer au client Android via l'interface mobile représente la liste des zones et des adresses MAC enregistrées. Encore une fois XML nous permet d'avoir une représentation complète des informations tout en étant très structuré et très simple à comprendre. Voici un exemple de XML qui pourrait être transmis :

```
<?xml version="1.0" encoding="utf-8" ?>
<zones>
<zone>
```

```
<name>Valais</name>
<latitude>62.12</latitude>
<longitude>51.1</longitude>
<dns>1.2.3.4</dns>
<addresses>
<address>
<mac>00-24-81-68-F4-05</mac>
<port>10120</port>
</address>
</addresses>
</zone>
<zone>
<name>Yverdon</name>
<latitude>63.32432</latitude>
<longitude>53.234234</longitude>
<dns>home.no-ip.org</dns>
<addresses>
<address>
<mac>00-24-81-68-F4-00</mac>
<port>12300</port>
</address>
<address>
<mac>00-24-81-68-F4-01</mac>
<port>12500</port>
</address>
</addresses>
</zone>
</zones>
```

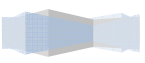
L'exemple ci-dessus contient deux zones différentes, une en Valais, une à Yverdon. Le XML pourrait bien en contenir une seule ou bien plus. Le même principe s'appliquer pour les ordinateurs à réveiller contenus dans '<addresses>'. La première zone en contient un seul, la seconde en contient deux. Il n'y a évidemment pas de limite (tout en restant dans un nombre convenable) à ces quantités. On peut également repérer la présence d'un nom de zone accompagné des coordonnées de cette zone. La balise '<dns>' quant à elle peut tout aussi bien représenter un nom de domaine qui sera résolu par Android (on peut tout à fait imaginer l'installation d'un dns dynamique tel que no-ip.org dans le réseau à réveiller pour contourner les problèmes de changement d'adresse IP) qu'une adresse IP réelle.

La balise '<addresses>' contient quant à elle deux informations. Si la 1ère est évidente pour les connaisseurs du Wake On Lan (il s'agit bien sûr de l'adresse MAC de l'ordinateur à réveiller), la 2ème information contient le port à utiliser pour la redirection dans le réseau local. En effet, pour que le Wake On Lan puisse passer du réseau Internet au réseau local d'un domicile, celui-ci doit traverser le routeur et être diffusé en broadcast local sur le réseau (192.168.1.255 par exemple). Ce port représente donc le port configuré sur le routeur qui sera redirigé vers un broadcast local.



Voici la DTD décrivant la structure de ce type de transmission XML :

```
<!DOCTYPE zones [
<!ELEMENT zones (zone*)>
<!ELEMENT zone (name,latitude,longitude,dns,addresses)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT latitude (#PCDATA)>
<!ELEMENT longitude (#PCDATA)>
<!ELEMENT dns (#PCDATA)>
<!ELEMENT addresses (address*)>
<!ELEMENT address (mac,port)>
<!ELEMENT mac (#PCDATA)>
<!ELEMENT port (#PCDATA)>
]>
```

6.2. Android

6.2.1. Activités et Service

Nous avons implémenté trois activités différentes et un service :

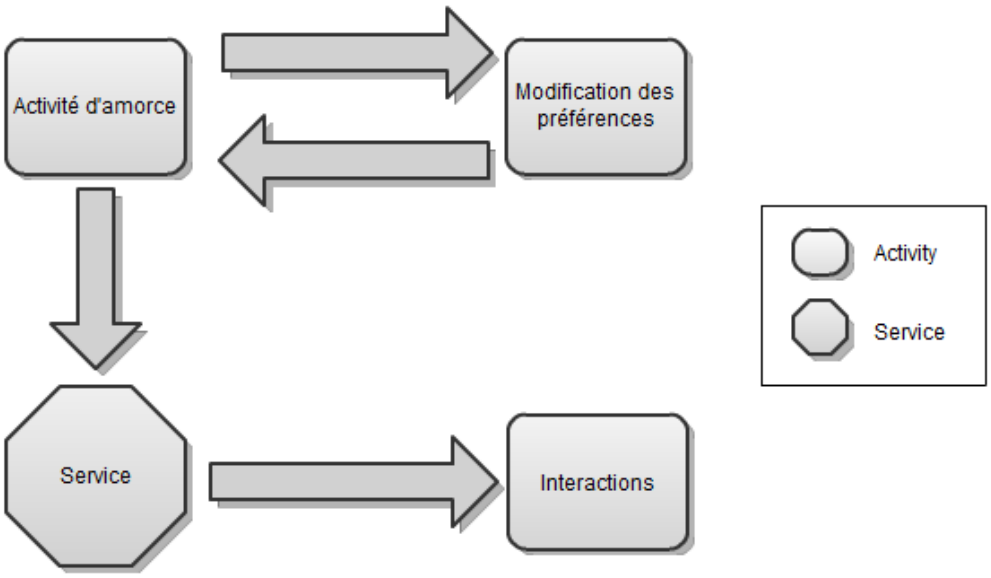


Figure 25 : Diagramme de représentation générale des Activités et Service du programme

Nous pouvons constater que nous avons une activité d'amorce. Quand le programme va être lancé, c'est cette activité qui va démarrer en premier. Cette activité va afficher l'état du programme et vérifier si l'utilisateur a déjà entré des données d'identification valides. Les données d'identification vont servir à s'authentifier sur le serveur afin de récupérer les zones d'intérêt de l'utilisateur et sont composées d'un e-mail ainsi que d'un mot de passe. Si l'utilisateur n'a toujours pas entré d'informations d'identification, notre activité principale va lancer une deuxième activité qui va servir d'interface avec l'utilisateur pour la saisie et la modification de ses données. Puis, quand des données valides seront entrées dans le programme, notre activité d'amorce va se connecter au serveur pour récupérer les points d'intérêt. Finalement, quand les points d'intérêt seront récupérés, l'activité d'amorce lancera notre service.

Notre service aura la tâche de s'abonner aux modifications d'emplacements de l'utilisateur. Donc notre service va recevoir périodiquement de nouvelles coordonnées concernant la position de l'utilisateur. A chaque fois qu'il reçoit de nouvelles coordonnées, notre service va les comparer aux coordonnées des points d'intérêts et va mettre à jour la liste des interactions disponibles pour l'utilisateur. Par exemple, si l'utilisateur se trouve chez lui ou à proximité de sa maison, il aura la possibilité d'allumer les ordinateurs de sa maison et non de tous les points d'intérêts qu'il a fixé, ceci pour pouvoir lui proposer plus facilement ce qu'il cherche à faire.

Pour que l'utilisateur puisse choisir si il veut ou non interagir avec un de ses points d'intérêt à proximité, le service va afficher un message dans la barre de statut de l'appareil :

Figure 26 : Icône dD de la barre de statuts

Nous pouvons voir que le service ajoute une icône dans la barre des statuts du téléphone.

Figure 27 : Notification d'interaction possible

Si l'utilisateur va regarder les notifications dans cette barre de statut, il va trouver ce message. En cliquant dessus il va être amené sur l'activité des interactions.

Figure 28 : Exemple d'interaction avec un point d'intérêt

Voici la présentation de l'activité des interactions possibles avec par exemple un point d'intérêt actif. Dans ce cas l'utilisateur peut allumer les ordinateurs présents dans le point d'intérêt « Maison ».

Voici un diagramme qui donne un peu plus de précisions concernant les différentes étapes importantes du fonctionnement de notre activité d'amorce :

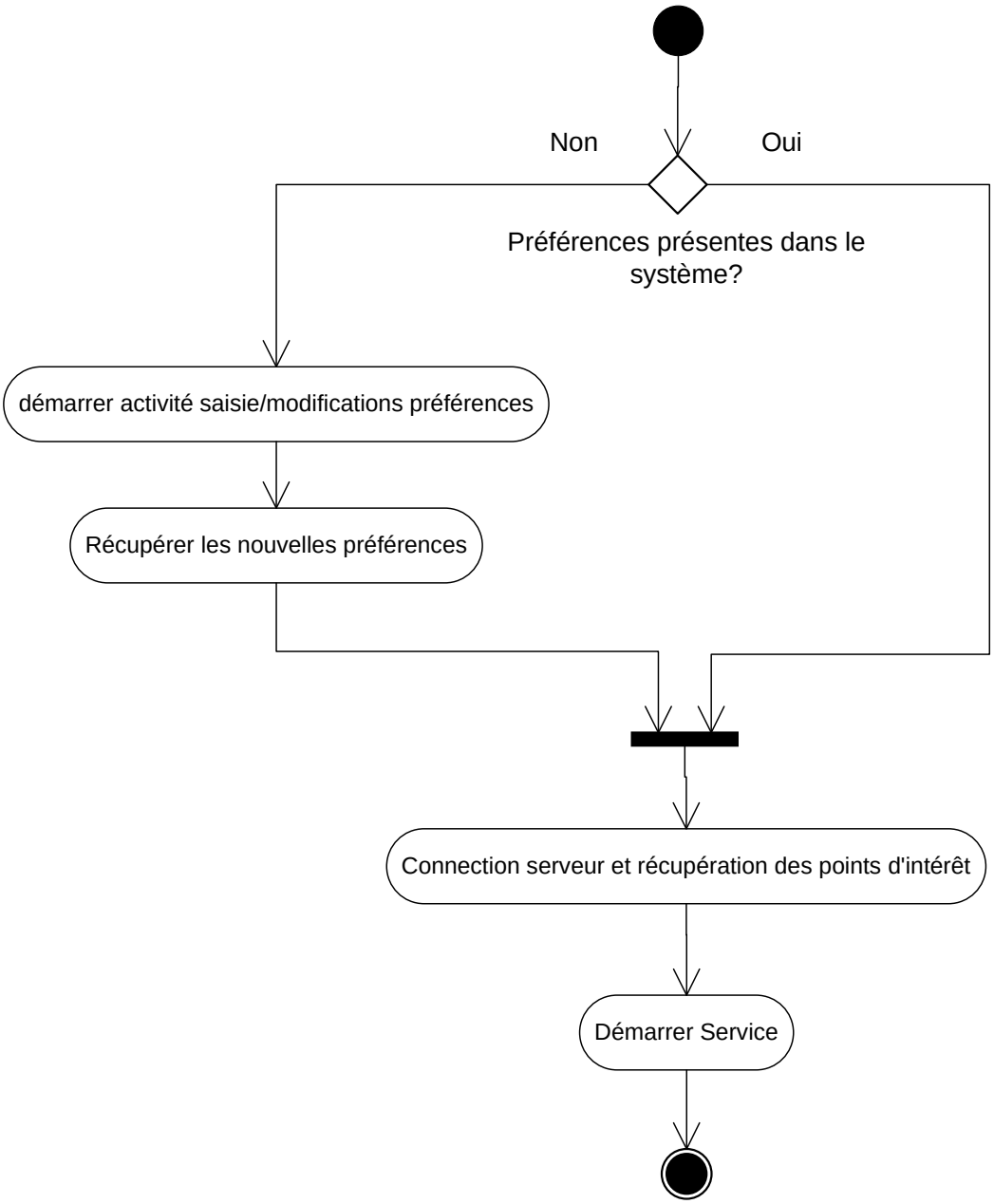


Figure 29 : Fonctionnement de l'activité d'amorce

Une fois le service démarré, notre utilisateur n'aura qu'à appuyer sur la touche retour pour laisser notre activité d'amorce en arrière plan et ainsi pouvoir disposer du téléphone pour toute autre tâche. Nous avons pensé implémenter de cette manière notre programme pour que l'utilisateur puisse profiter pleinement de son Smartphone pendant que notre application fonctionne en arrière plan grâce à notre service.

Nous avons dû réfléchir à deux problèmes importants :

1) Positionnement par GPS ou par réseau ?
   Pour répondre à cette question, nous avons dû tester si le GPS était utile ou non pour l'utilisateur tout en sachant qu'il est extrêmement gourmand en énergie. Nous pensions tout simplement allumer le GPS pour avoir des coordonnées plus précises une fois que la personne s'approchait du point d'intérêt. Nous avons donc fait quelques essais et avons constaté que le GPS n'était pas essentiel pour le bon fonctionnement de notre programme car :
   - Il consomme beaucoup d'énergie
   - Si la personne arrive à la maison en voiture, le GPS a besoin de trop de temps pour commencer à recevoir des positions précises. Lors de nos tests, nous somme arrivés vers un point d'intérêt et le système n'a détecté notre position que 10 minutes après car nous n'avions pas de réception GPS ni dans la voiture ni dans la maison du point d'intérêt.
   - Finalement, même si le positionnement par le réseau téléphonique est moins précis, il reste tout à fait utilisable.

   Donc nous avons pris la décision de ne pas allumer le GPS à proximité du point d'intérêt car cela était inutile et gourmand en énergie. Notre positionnement se fait via le réseau téléphonique qui a une précision qui varie en moyenne de 100 m à 2 km. Ceci est largement suffisant car nous voulons tout simplement détecter si la personne est à proximité d'un point d'intérêt pour lui permettre d'allumer ses ordinateurs.

2) Allumer le Wifi pour faire un Wake On Lan ?
   Encore une fois, pour cette question nous avons dû nous pencher sur les avantages et les inconvénients d'allumer automatiquement le Wifi. Nous avons fait des essais et encore une fois, l'allumage automatique du Wifi s'est révélé inutile pour notre programme car :
   - Il est gourmand en énergie et l'utilisateur n'en a pas forcément besoin
   - Nous avons implémenté la possibilité de faire un Wake On Lan à travers le Wifi mais aussi à travers le réseau 3G. Ceci sera expliqué plus loin dans cette documentation.

   Sachant que la personne n'a pas forcément besoin d'allumer le Wifi et qu'elle possède un bouton de raccourcis pour le faire dans notre programme, nous avons décidé de ne pas l'allumer automatiquement.

Maintenant qu'une idée générale du fonctionnement de l'application est donnée, voici les différentes vues affichées par les activités :

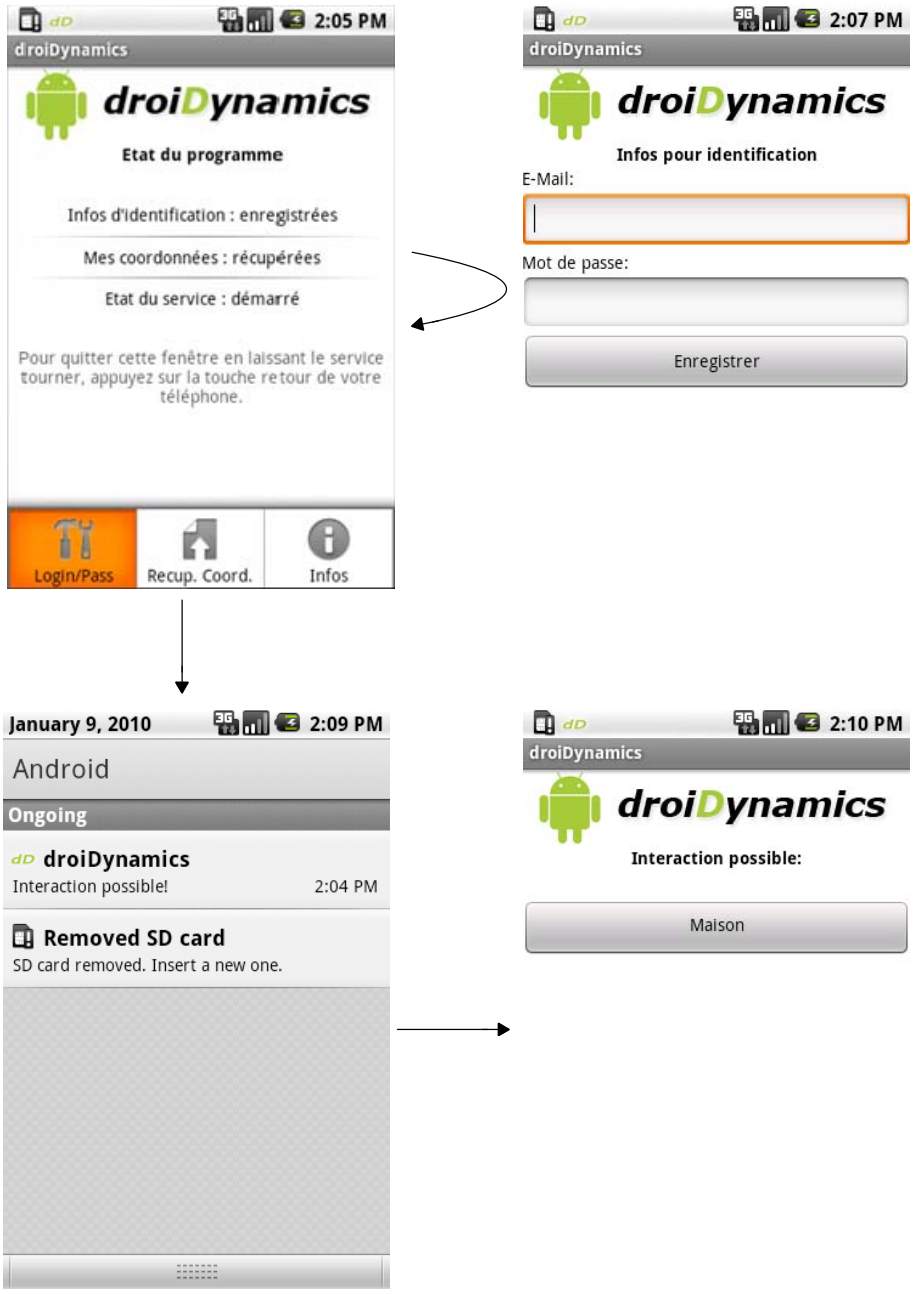


Figure 30 : Les différentes vues de notre programme

Nous pouvons faire la corrélation entre ce graphique et celui des activités et service de notre application qui vous a été présenté plus haut.

En haut à gauche nous pouvons voir l'interface de notre activité d'amorce, qui appelle l'activité de saisie/modification des préférences, puis notre service en bas à gauche et finalement l'activité de gestion des interactions en bas à droite. Chaque flèche correspond à un appel et la flèche avec retour, signifie que l'activité principale fait appel à notre activité de saisie des préférences en lui demandant en retour les valeurs qu'a entré l'utilisateur.

6.2.2. Wake On Lan

Le « Wake On Lan » veut dire « réveil via réseau ». Pour simplifier, au lieu de démarrer un ordinateur en appuyant sur un bouton, nous allons tout simplement l'allumer à travers le réseau Ethernet. Pour cela, il nous faut lui envoyer une séquence spéciale sur le port 9 (anciennement port 0 ou aussi port 7) en utilisant l'adresse de broadcast.

Par exemple, si nous voulons allumer un ordinateur possédant l'adresse mac xx :xx :xx :xx :xx et l'adresse IP 192.168.1.2, il nous faudra envoyer une trame sur l'adresse 192.168.1.255 ou 255.255.255.255 avec comme port numéro 9 de destination. Cette trame devra être composée de la manière suivante :

On commence par les octets FF FF FF FF FF FF suivis de seize répétitions de l'adresse MAC de la cible, puis d'un mot de passe (si nécessaire) de quatre ou six octets.

Il est possible de lancer un Wake On LAN à travers Internet, vers une machine située derrière un routeur NAT[17], mais ceci sous certaines conditions : le paquet magique doit être un paquet UDP, dont le port utilisé est redirigé vers l'adresse IP de la machine qui doit être réveillée.[18]

Dans notre programme, nous laissons libre choix à l'utilisateur de paramétrer l'ensemble adresse mac, adresse Ip et port qu'il veut pour faire soit du Wake On Lan via le Wifi, doit du Wake On Lan via Internet (3G).

[17] En informatique, on dit qu'un routeur fait du Network Address Translation (NAT) lorsqu'il fait correspondre les adresses IP internes non-uniques et souvent non routables d'un intranet à un ensemble d'adresses externes uniques et routables. Source : http://fr.wikipedia.org/wiki/Network_address_translation
[18] http://fr.wikipedia.org/wiki/Wake-on-LAN#iPhone

7. Le code

7.1. Serveur Web

La partie Web consiste en trois parties. La base de données qui regroupe les informations sur les utilisateurs et leurs zones et les deux sites présentés dans le fonctionnement général.

7.1.1. Base de données

La base de données est évidemment un des piliers du projet. Sans elle, l'enregistrement serait impossible, du moins pas en conservant une propreté et une clarté des informations. En effet, les informations auraient pues être enregistrées dans des fichiers textes par exemple, mais le traitement de ces fichiers aurait été bien plus lourd qu'avec une base de données sur laquelle on peut faire des requêtes SQL.

La base de données que nous avons conçue est composée des tables suivantes :

7.1.1.1. Table imei

La table imei contient simplement les codes imei hachés des utilisateurs ainsi que leur description. Elle est composée des champs suivants :

- Id (type entier) : Numéro unique identifiant le code IMEI dans la table (clé primaire). Nous avons décidé de ne pas fixer la clé primaire sur le code IMEI en lui-même, afin qu'un code puisse être utilisé par deux comptes.
- Code (type char, longueur 128) : Ce champ représente le code IMEI haché. Etant donné que l'algorithme utilisée est le SHA512, l'emprunte fera toujours 128 caractères hexadécimaux.
- Name (type char, longueur variable) : Ce champ est utilisé pour enregistrer la description d'un code IMEI de l'utilisateur.
- User (type entier) : Ce champ spécifie à quel utilisateur est lié le code IMEI. Il permet donc de réaliser la relation entre les tables imei et user.

7.1.1.2. Table language

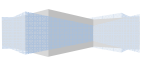
La table language n'est pas utilisée dans la version actuelle de l'application. Elle permet de stocker la liste des langues dans lesquelles le site serait disponible. Cette table est composée des champs suivants :

- Id (type entier) : Numéro identifiant la langue dans le système (clé primaire).
- Name_short (type char, longueur variable) : Ce champ permet de stocker un nom court pour la langue ou une abbréviation.
- Name_long (type char, longueur variable) : Ce champ permet de stocker le nom complet de la langue.

7.1.1.3. Table user

La table user permet de stocker la liste des utilisateurs inscrits au site (donc à l'application). C'est une des tables centrales de la base de données, puisqu'elle est liée à la table des zones. Les champs suivants la caractérisent :

- Id (type entier) : Numéro identifiant l'utilisateur dans le système (clé primaire).



- Mail (type char, longueur variable) : C'est dans ce champ qu'est enregistrée l'adresse mail de l'utilisateur pour l'identification.
- Password (type char, longueur : 128) : Ce champ permet de stocker le mot de passe haché de l'utilisateur. Puisque c'est l'algorithme SHA512 qui est utilisé pour réaliser cette emprunte, le mot de passe fait toujours 128 caractères.
- Language (type entier) : Ce champ permet de définir la langue de l'utilisateur. Il est en relation avec la table language.

7.1.1.4. Table wol_address

Cette table contient la liste des adresses à réveiller pour chaque zone. Elle possède les champs suivants :

- Id (type entier) : Numéro identifiant l'adresse dans le système (clé primaire).
- Mac (type char, longueur 17) : Ce champ contient l'adresse MAC. Il est représenté par 17 caractères car une adresse MAC est composée de 5 caractères de séparation et de 6 blocs de 2 caractères.
- Port (type entier) : C'est le numéro de port à utiliser pour entrer dans le réseau de cette adresse MAC.
- Zone (type entier) : Ce champ permet de définir à quelle zone appartient l'adresse MAC. Il permet la relation entre la table zone et la table wol_address.

7.1.1.5. Table zone

Cette table contient la liste des zones des utilisateurs avec leurs coordonnées. Elle est définie avec les champs suivants :

- Id (type entier) : C'est le numéro identifiant la zone dans le système (clé primaire).
- Latitude (type double) : Ce champ contient la 1ère coordonnée de localisation de la zone.
- Longitude (type double) : Ce champ contient la 2ème coordonnée de localisation de la zone.
- Name (type char, longueur variable) : Ce champ contient le nom décrivant la zone.
- Dns (type char, longueur variable) : Ce champ contient l'adresse à laquelle envoyer les Wake on Lan pour cette zone.
- User (type entier) : Ce champ permet la liaison avec la table user afin de définir quel utilisateur à créé cette zone.

7.1.2. Site complet

Le site est constitué d'une certaine quantité de fichiers qui peut être difficile à comprendre sans connaître à quoi ils servent. Cette partie décrira donc l'utilité de chaque élément dans l'arborescence du site. Le fonctionnement précis n'est pas expliqué ici étant donné que le code source est intensivement commenté.

7.1.2.1. Répertoire /images/

Ce dossier contient les différentes images utilisées dans le site Web ainsi que leur source.

- /images/icon.png : Cette image représente une zone sur la carte Google Maps. Elle est représentée par un petit robot faisant un signe du bras.
- /images/icon.psd : C'est le fichier source Adobe Photoshop de l'image précédente.

- /images/index.php : Ce fichier PHP est vide, il sert simplement à ne pas afficher le contenu du dossier si un visiteur essaie de s'y connecter.
- /images/logo.png : C'est le logo du site internet et du projet. Cette image est affichée au sommet des pages du site.
- /images/logo.psd : C'est le fichier source Adobe Photoshop de l'image précédente.
- /images/logo_login.png : C'est le logo que vous retrouvez sur la page d'identification et d'enregistrement.
- /images/logo_login.psd : C'est le fichier source Adobe Photoshop de l'image précédente.

7.1.2.2. Répertoire /includes/

Ce dossier contient différents scripts PHP qui sont inclus dans le fichier principal pour lui ajouter des fonctionnalités.

- /includes/etc.php : Ce fichier contient des fonctions PHP diverses qui ne trouvaient pas place dans les autres fichiers. Les fonctions suivantes sont fournies par ce fichier :
  - check_string : Fonction permettant de contrôler la validité d'une chaîne de caractères sur le principe de la liste blanche grâce aux expressions régulières.
  - check_mail : Fonction permettant de contrôler la validité potentielle d'une adresse mail (caractères uniquement) grâce aux expressions régulières. Elle contrôle donc que la partie avant le '@' soit composée d'au moins un caractère, nombre, '.', '_' ou '-'. Elle vérifie ensuite la présence d'un '@' puis la présence d'autre moins deux caractères, nombres, '.', '_' ou '-' suivi d'un '.' et d'une extension de deux à quatre caractères.
- /includes/html.php : Le fichier html.php contient les fonctions qui vont retourner du code HTML à afficher au fichier principal du site :
  - html_get_login : Fonction permettant de créer le code HTML de la page de login du site.
  - html_get_register : Fonction permettant de créer le code HTML de la page d'enregistrement du site.
  - html_get_home : Fonction permettant de créer le code HTML de la page d'accueil du site.
  - html_get_password : Fonction permettant de créer le code HTML de la page de changement de mot de passe et d'imei.
  - html_get_zone : Fonction permettant de créer le code HTML de la page de modification d'une zone.
  - html_get_layout_start : Fonction permettant de créer le code HTML du début de la page (appelée avant les fonctions décrites ci-dessus, sauf pour html_get_login et html_get_register qui sont « externes » au site).
  - html_get_layout_end : Fonction permettant de créer le code HTML de la fin de la page (appelée après les fonctions décrites ci-dessus, sauf pour html_get_login et html_get_register qui sont « externes » au site).
- /includes/index.php : Ce fichier PHP est vide, il sert simplement à ne pas afficher le contenu du dossier si un visiteur essaie de s'y connecter.
- /includes/mysql.php : Ce fichier contient les fonctions ayant un lien avec la base de données :

  - mysql_connection : Fonction permettant de se connecter à la base de données MySQL en configurant l'encodage de la connexion en utf8.
  - mysql_protect_string : Fonction permettant de protéger une chaine de caractères pour une requête MySQL (par exemple ' deviendrait \')
- /includes/user.php : Ce fichier contient les fonctions liées à la gestion des utilisateurs :
  - user_get_id : Fonction permettant d'authentifier un utilisateur sur le site en lui créant une session si il vient de s'identifier, ou de lui prolonger sa session si il a juste changé de page.
  - user_logout : Fonction permettant de déconnecter un utilisateur en effaçant sa session sur le serveur et son cookie de session local.
  - user_get_ip : Fonction permettant de récupérer l'adresse Ip du visiteur. Nous n'avons pas écrit cette fonction, elle provient du site officiel de PHP : http://www.php.net.

7.1.2.3. Répertoire /scripts/

Ce dossier contient les scripts JavaScript nécessaires à l'exécution du site sur le navigateur de l'utilisateur.

- /scripts/index.php : Ce fichier PHP est vide, il sert simplement à ne pas afficher le contenu du dossier si un visiteur essaie de s'y connecter.
- /scripts/login.js : Ce fichier contient la fonction qui est appelée avant l'envoi du formulaire d'identification de l'utilisateur :
  - createPasswordLogin : Fonction permettant d'ajouter un sel à un mot de passe et de le hacher pour l'envoi au serveur.
- /scripts/maps.js : Ce fichier contient les fonctions JavaScript que nous avons écrites afin d'interagir avec l'API de Google Maps :
  - removeTemp : Fonction permettant de supprimer un marker et une InfoWindow temporaire (affichés pour l'ajout d'une zone).
  - getAddHtml : Fonction permettant d'obtenir le code html à insérer dans une InfoWindow d'ajout de zone.
  - addMarker : Fonction permettant d'ajouter un marker à la carte.
  - handleRightClick : Fonction permettant de gérer le clic droit (pour ajouter une zone).
  - googleMapsInit : Fonction permettant de configurer et d'initialiser la carte Google Maps.
- /scripts/sha512.js : Ce fichier implémente l'algorithme de SHA512 en JavaScript qui nous est utile pour notre protocole de cryptage des données sensibles. Nous ne sommes pas les auteurs de ce code, il peut être trouvé à l'adresse suivante : http://pajhome.org.uk/crypt/md5

7.1.2.4. Répertoire /styles/

Ce dossier contient les fichiers de styles CSS pour le rendu visuel du site.

- /styles/index.php : Ce fichier PHP est vide, il sert simplement à ne pas afficher le contenu du dossier si un visiteur essaie de s'y connecter.

- /styles/layout.css : Ce fichier contient les différentes propriétés CSS pour les pages du site Internet.
- /styles/map.css : Ce fichier contient les styles qui s'appliquent à la carte Google Maps.

7.1.2.5. Répertoire /

Ce dossier contient la page principale du site ainsi que certains éléments conseillés.

- /favicon.ico : Ce fichier est vide. Nous l'avons ajouté étant donné que les navigateurs essaient de le récupérer à chaque visite sur le site. Si ce fichier n'existe pas, cela va générer une ligne de log d'erreur sur le serveur. Cependant, ce fichier n'influence pas du tout le fonctionnement du site.
- /index.php : C'est le fichier principal du site. En fonction de l'état du visiteur (déconnecté ou connecté en tant qu'utilisateur) et des paramètres qui lui sont passés (index.php ?page=PAGE), il va récupérer du code HTML dans les fonctions du fichier /includes/html.php et l'afficher.
- /robots.txt : Ce fichier n'a aucune influence non plus sur le fonctionnement du site. Cependant, lorsque les robots des moteurs de recherche scannent le site, l'absence de ce fichier de consignes provoque également l'ajout d'une ligne de log sur le serveur.

7.1.3. Site mobile

Le site mobile est un site allégé (pour des connexions lentes) qui se contente d'envoyer les zones de l'utilisateur lorsqu'il est authentifié. Le code est commenté de façon complète dans les fichiers sources.

Le site mobile est constitué des éléments suivants :

7.1.3.1. Répertoire /includes/

Ce dossier contient différents scripts PHP qui sont inclus dans le fichier principal pour lui ajouter des fonctionnalités comme pour le site principal. De plus, les trois fichiers présents dans ce dossier (etc.php ,index.php et mysql.php) sont les mêmes que ceux du sites principal. Pour plus d'information, il suffit de se référer au chapitre précédent.

7.1.3.2. Répertoire /

Comme pour le site principal, ce dossier contient un fichier robots.txt et favicon.ico. Ces fichiers sont présents pour les mêmes raisons que sur l'autre site.

Un fichier index.php est également présent, mais celui-ci est en revanche complètement différent. Dans cette version, on commence par contrôler si on a envoyé un token à l'utilisateur. Si c'est le cas, on contrôle si celui-ci nous a transmis ses informations d'identification. Si en revanche aucun token n'a été envoyé à l'utilisateur, on en génère un et on le lui transmet.

Si toutes les informations d'identification sont reçues, on commence par contrôler si l'adresse mail est correcte et si elle existe dans la base de données. Si ce n'est pas le cas, on retourne une erreur à l'utilisateur.

On vérifie ensuite le mot de passe associé à l'adresse mail et finalement si l'IMEI reçu est dans la liste des IMEI autorisés par l'utilisateur. Si toutes ces informations sont valides, on transmet à l'utilisateur le fichier XML contenant ses zones.

7.2. Partie Android

Pour avoir une vue d'ensemble des classes utilisées dans le code, voici les diagrammes de classe.

7.2.1. Partie concernant l'activité d'amorce

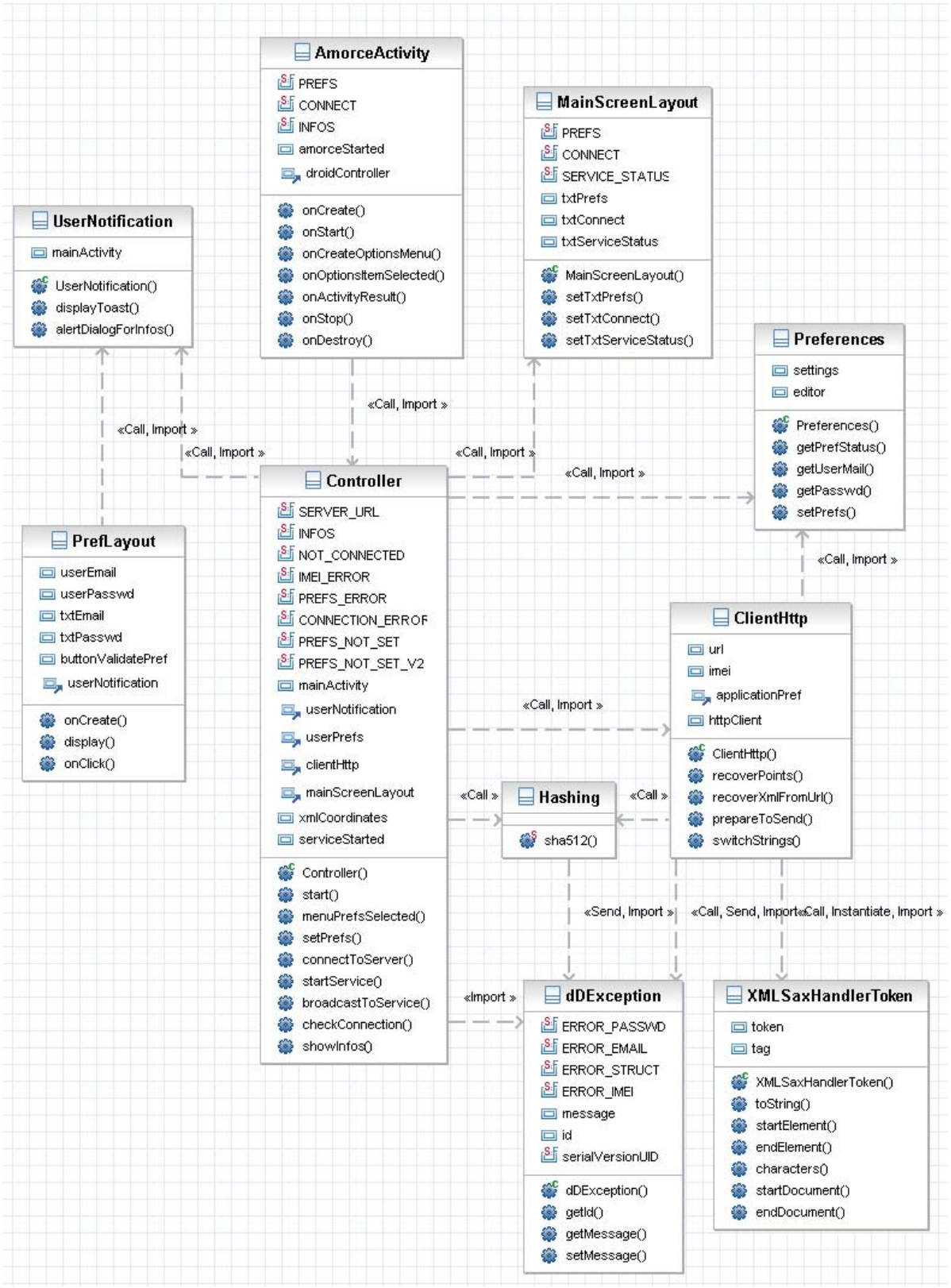


Figure 31 : Diagramme de classes, partie activité d'amorce

Nous pouvons constater que la classe centrale est la classe « Controller ». C'est elle qui gère tous les éléments de cette première partie du programme.

Nous allons détailler les classes une par une pour mieux comprendre leur fonctionnement tout en terminant par le Controller pour voir comment se mettent en places les liens entre nos classes :

7.2.1.1. Hashing

Figure 32 : Classe Hashing

Nous utilisons cette classe pour hacher une chaîne de caractères en SHA512. Ceci va nous retourner une autre chaîne de caractères avec la réponse du hach. La seule fonction présente (sha512) est statique et publique.

7.2.1.2. dDException

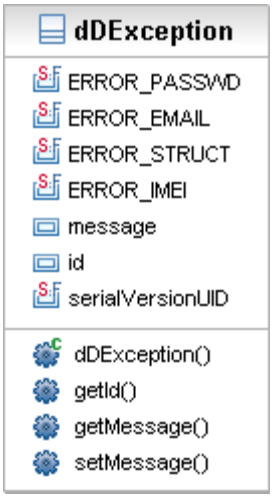


Figure 33 : Classe dDException

Cette classe nous permet de gérer des erreurs personnalisées. Elle n'est pas essentielle en soit car on peut tout simplement se baser sur la classe Exception, mais nous avons choisit de gérer plus facilement les erreurs avec cette nouvelle classe.
Un des avantages est que nous retrouvons des chaînes de caractères constantes et publiques pour gérer les erreurs XML reçues par le serveur.
Cette classe ne sera utilisée pratiquement que par notre ClientHTTP.
Une erreur est définie par un id et un message. De plus cette classe est étendue de la classe Exception.

7.2.1.3. Preferences

Figure 34 : Classe Preferences

Dans cette classe nous gérons les préférences. Il faut savoir qu'Android nous met à disposition un outil simplifié pour pouvoir sauvegarder nos préférences d'application facilement dans le système. De cette manière à chaque fois que l'application est lancée elle récupère les préférences sauvegardées de la dernière fois. Ceci pour que l'utilisateur ne doivent pas les entrer à chaque fois.
Les préférences sont liées à notre activité d'amorce et sont composées d'un email et d'un mot de passe haché avec du SHA512.

7.2.1.4. XMLSaxHandlerToken

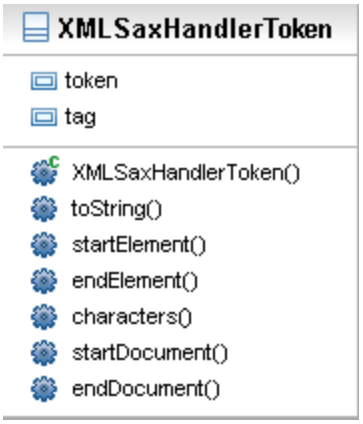


Figure 35 : Classe XMLSacHandlerToken

Cette classe gère un parser SAX. Nous avons les fonctions classiques de début/fin de document, début/fin de balise et lorsqu'on récupère le texte d'une balise.
Cette classe gère la récupération du jeton qui nous est fourni par le serveur lors de notre première connexion. Elle est utilisée depuis la classe ClientHTTP et retourne la valeur via sa fonction toString.

7.2.1.5. ClientHTTP

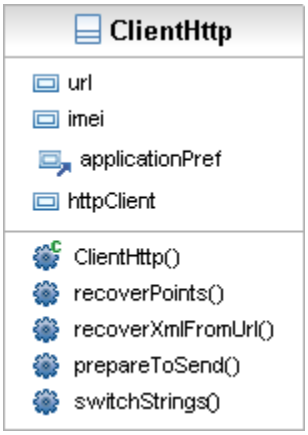


Figure 36 : Classe ClientHTTP

Cette classe est très importante car c'est elle qui va faire la liaison avec le serveur pour y récupérer les différentes coordonnées des points d'intérêt.
Pour cela elle a besoin d'une liaison à notre classe Preferences.
Le controller va appeler la fonction recoverPoints qui va retourner la réponse du serveur en XML. Pour cela il y a plusieurs étapes à faire :

1) Récupérer un jeton du serveur en faisant tout simplement un get de l'url : http://mobile.droidynamics.mobi/index.php
2) Parser le résultat XML pour avoir la valeur du jeton grâce à la classe XMLSaxHandlerToken
3) Envoyer une demande des points d'intérêt avec les préférences et l'IMEI de l'appareil de l'utilisateur de la manière suivante :

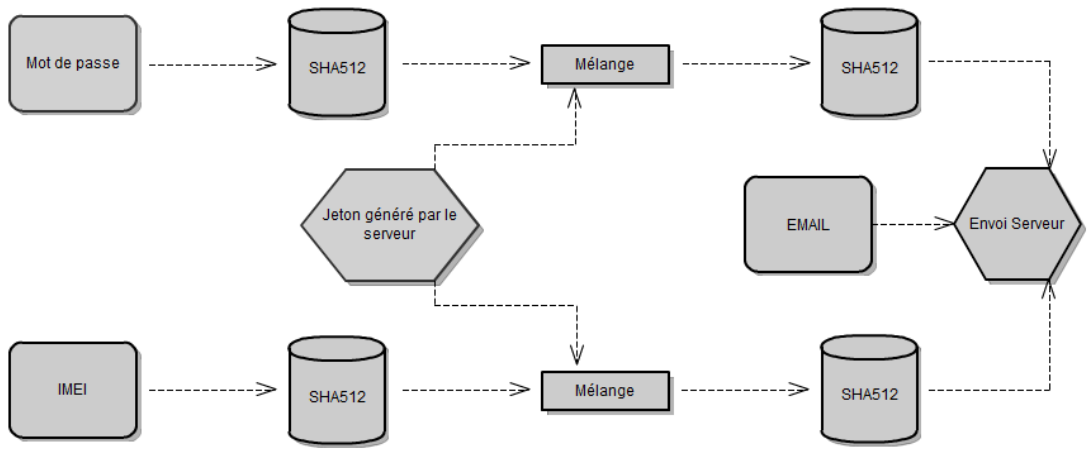


Figure 37 : Génération de la demande des zones

4) Une fois le XML avec les points d'intérêts récupéré, on renvoie le résultat au Controller.

7.2.1.6. UserNotification

Figure 38 : Classe UserNotification

Cette classe nous permet de gérer les notifications qui vont être faite à l'écran pour l'utilisateur.

Il y a deux notifications possibles :

1) Un toast : petit message qui s'affiche en « info bulle » vers le bas de l'écran
2) Une boite de dialogue qui apparaît sur l'écran de l'utilisateur

Voici deux images qui illustrent ces deux types de notifications :

Figure 39 : Toast

Figure 40 : Boîte de dialogue

7.2.1.7. PrefLayout

Figure 41 : Classe PrefLayout

Cette classe gère l'affichage et la gestion de la vue concernant la saisie/modification des préférences de l'utilisateur. Pour créer une vue, il y a deux manières de faire :

1) Le faire via le code en ajoutant élément par élément les objets dans notre layout
2) A l'aide d'un fichier XML de layout et en récupérant les éléments en utilisant leur id via la classe R (prégénérée dans le SDK Android)

Concernant le fichier XML, voici le code de présentation :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffff">
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/img_pref_top"
    android:layout_gravity="center"
    android:src="@drawable/droidynamics_top_image" />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/str_pref_infos"
    android:textColor="#000000"
    android:gravity="center_horizontal"
    android:textStyle="bold"/>
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/str_pref_email"
    android:textColor="#000000"
    android:id="@+id/txt_pref_email"/>
<EditText
    android:id="@+id/etxt_pref_email"
    android:maxLines="1"
    android:layout_marginTop="2dip"
    android:layout_width="fill_parent"
    android:ems="25"
    android:layout_height="wrap_content"
    android:scrollHorizontally="true"
    android:autoText="false"
    android:maxLength="255"
    android:singleLine="true"/>
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
    android:text="@string/str_pref_passwd"
    android:textColor="#000000"
    android:id="@+id/txt_pref_passwd"/>
<EditText
    android:id="@+id/etxt_pref_passwd"
    android:maxLines="1"
    android:layout_marginTop="2dip"
    android:layout_width="fill_parent"
    android:ems="25"
    android:layout_height="wrap_content"
    android:scrollHorizontally="true"
    android:password="true"
    android:autoText="false"
    android:singleLine="true"
    android:maxLength="255"/>
<Button
    android:id="@+id/btn_pref_validate"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/str_pref_btn_validate"/>
</LinearLayout>
```

Ce qu'il faut retenir de ce code est que pour créer un layout il nous faut la balise « LinearLayout » puis créer à la suite les différents éléments graphiques. Nous retrouvons les éléments suivants :

- ImageView : représente une image, dans notre cas ce sera le logo droiDynamics
- TextView : représente un label
- EditText : représente un champ de saisie de texte
- Button : représente un bouton

Tous ses éléments ont des attributs qui les définissent et qui sont simples à comprendre, mais un attribut est tout particulièrement important : « android:id ». Ceci car dans le notre code, pour interagir avec un élément graphique, nous allons aller le chercher via son id :

Button myButton = (**Button**) this.**findViewById**(**R.id.**btn_pref_validate);

Voici le résultat de la vue saisie/modification des préférences:

Figure 42 : Interface de saisie des préférences

L'utilisateur va tout simplement saisir son email ainsi que son mot de passe et appuyer sur le bouton « Enregistrer ».

Nous avons géré les cas où le serveur nous informe que l'une ou l'autre de ses informations sont erronées. Voici le résultat de l'affichage en cas d'erreur :

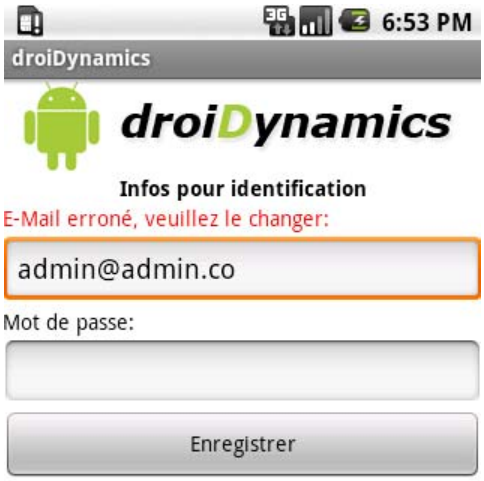


Figure 43 : Email erroné

Figure 44 : Mot de passe erroné

7.2.1.8. MainScreenLayout

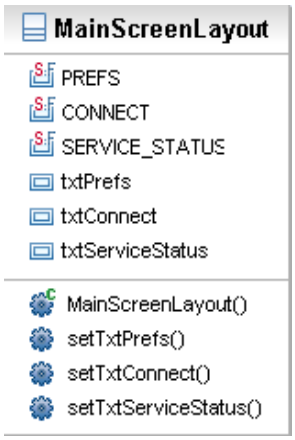


Figure 45 : Classe MainScreenLayout

Cette classe s'occupe de gérer la vue principale de la classe d'amorce qui affiche l'état du logiciel.
De plus, elle intègre un menu qui est géré dans la classe AmorceActivity.

Voici la présentation de l'écran :

7:02 PM
droiDynamics
droiDynamics
Etat du programme
Infos d'identification : enregistrées
Mes coordonnées : récupérées
Etat du service : démarré
Pour quitter cette fenêtre en laissant le service tourner, appuyez sur la touche retour de votre téléphone.

Figure 46 : Layout principal sans erreur

7:01 PM
droiDynamics
droiDynamics
Etat du programme
Infos d'identification : E-Mail erroné
Mes coordonnées : non récupérées
Etat du service : non démarré
Pour quitter cette fenêtre en laissant le service tourner, appuyez sur la touche retour de votre téléphone.

Figure 47 : Layout principal avec exemple d'erreurs

Voici le menu qui est affiché :

Figure 48 : Menu de l'activité d'amorce

Si l'utilisateur appuie sur le bouton « Login/Pass », il sera redirigé automatiquement sur l'activité de saisie/modification des préférences. S'il appuie sur le bouton « Recup. Coord. » une nouvelle connexion sera exécutée pour récupérer les coordonnées des points d'intérêt stockées sur le serveur web. Si, finalement, il appuie sur le bouton « Infos », le message suivant lui sera affiché :

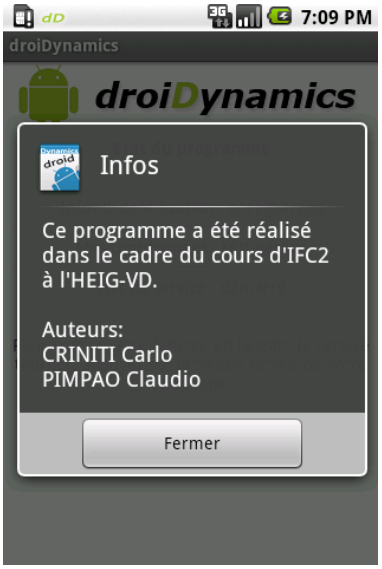


Figure 49 : Dialogue d'informations

Au niveau de la partie code, l'interaction avec le menu est gérée par la classe AmorceActivity et l'exécution des tâches est confiée à la classe Controller. Notre classe MainScreenLayout ne gère que l'affichage des infos à l'écran.

7.2.1.9. AmorceActivity

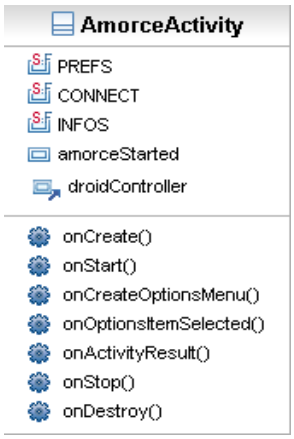


Figure 50 : Classe AmorceActivity

Cette classe gère la première activité qui va être démarrée par le système. Elle va tout simplement créer le « Controller » qui va se charger d'accomplir toutes les tâches de contrôle des préférences ainsi que la récupération des coordonnées des points d'intérêt et finalement la création du service.

De plus notre activité d'amorce implémente les fonctions onCreateOptionsMenu et onOptionsItemSelected qui vont gérer la création et le clic sur le menu.

7.2.1.10. Controller

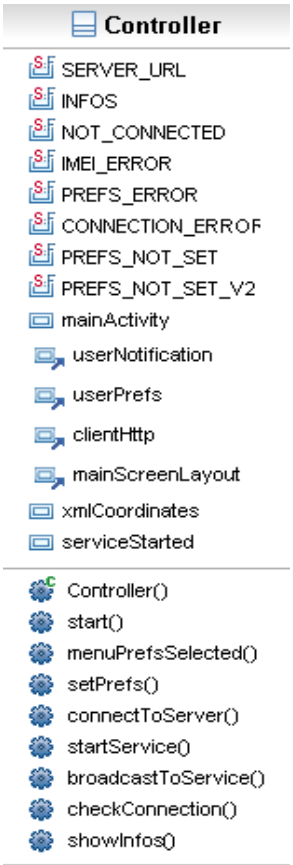
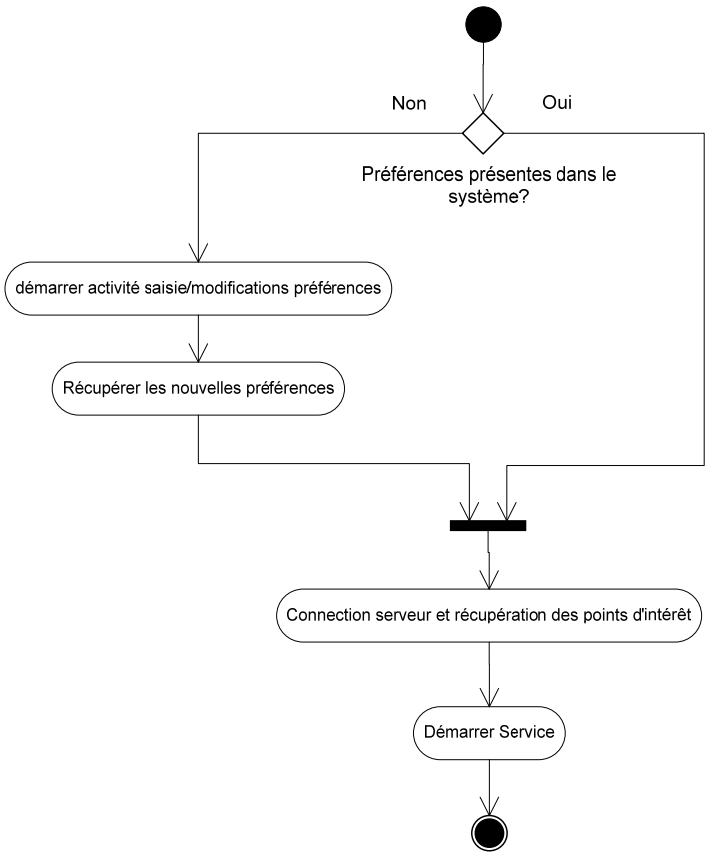


Figure 51 : Classe Controller

Dans cette classe, nous allons gérer la vue et toutes les actions qui vont se passer au début de notre application. Voici la liste de ce que fait notre controller :



Remarque *Nous avons la même image que plus tôt dans le document car, dans la partie explicative du rapport, nous n'avions pas encore introduit la classe du Controller, mais avions tout simplement parlé de l'activité d'amorce.*

Nous pouvons aussi noter que lors de la récupération du XML contenant les coordonnées des points d'intérêt de l'utilisateur, notre classe Controller, va vérifier que le serveur ne lui a pas répondu avec une erreur. Voici les codes XML d'erreurs possibles :

- <error>password</error> : le mot de passe entré par l'utilisateur est erroné
- <error>mail</error> : l'email entré par l'utilisateur n'est pas valide
- <error>structure</error> : la structure de ce qui a été envoyé au serveur n'est pas correcte
- <error>imei</error> : l'IMEI de notre appareil n'est pas enregistré sur le site

Voici les lignes concernant la création de notre Service :

```
Intent serviceIntent = new Intent(DroidService.EVENT_ACTION);
serviceIntent.putExtra("coordinates", xmlCoordinates);
mainActivity.sendBroadcast(serviceIntent);
```

Nous pouvons voir que la notion d'Intent est utilisée en passant l'action que l'on veut faire, et nous mettons des données en paramètres. Ces données sont tout simplement le XML des coordonnées des points d'intérêt de l'utilisateur.

7.2.2. Partie Service de l'application :

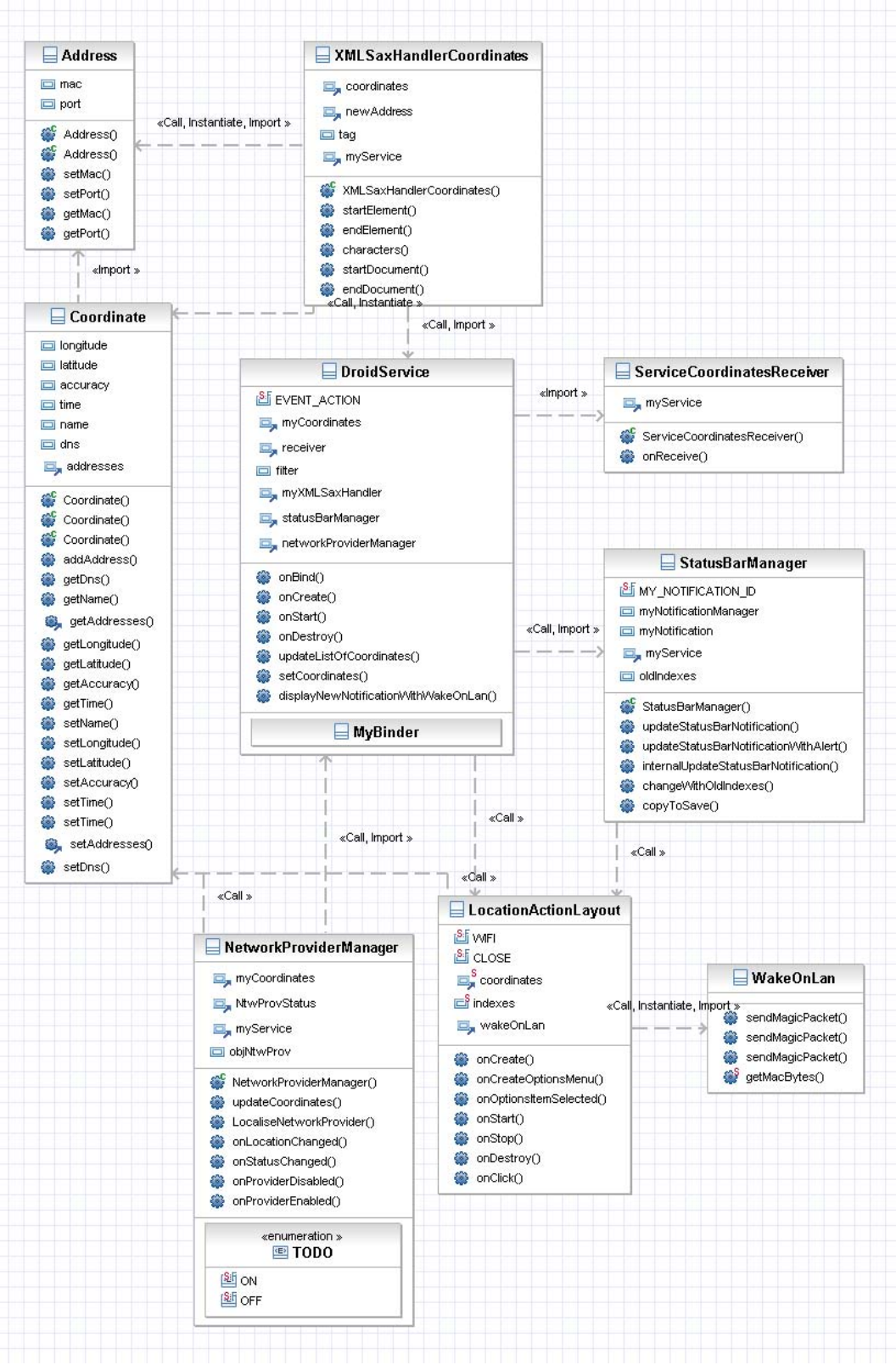


Figure 52 : Diagramme de classes, partie Service

7.2.2.1. Address

Figure 53 : Classe Address

Cette classe gère le stockage d'une adresse utile pour le Wake On Lan. Elle se compose d'une adresse Mac et d'un port. Concernant l'adresse Ip, elle est stockée dans la classe Coordinate car elle est commune pour tous les ordinateurs vu que c'est une adresse de broadcast.

7.2.2.2. Coordinate

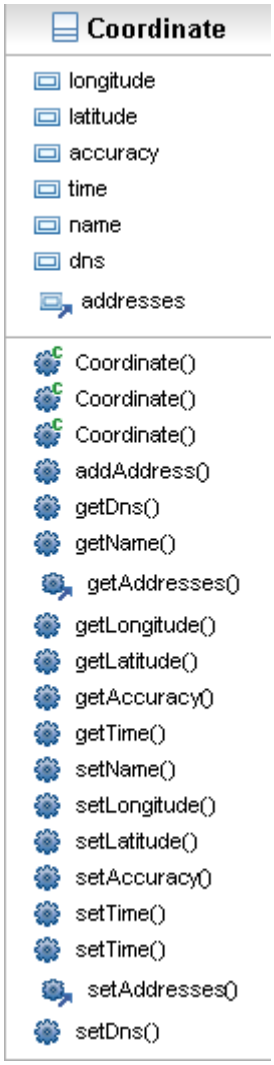


Figure 54 : Classe Coordinate

Cette classe va contenir une coordonnée complète, ce qui signifie : toutes les informations liées à un point d'intérêt. C'est en une liste des coordonnées que le XML contenant les points d'intérêt va être transformé (transformation faite pas le parser SAX de la page suivante dans la partie précédente).
Nous retrouvons les informations suivantes :

- La longitude
- La latitude
- La précision en mètres
- L'heure à laquelle nous avons créé cette coordonnée (cet attribut n'est pas utilisé dans notre code)
- Un nom qui défini le point d'intérêt (exemple : Maison)
- Un DNS ou une adresse Ip (car depuis l'extérieur l'utilisateur peut aussi avoir un adresse noip par exemple)
- Et finalement, une liste d'adresses qui correspondent à la paire Mac/port de chaque ordinateur.

7.2.2.3. XMLSaxHandlerCoordinates

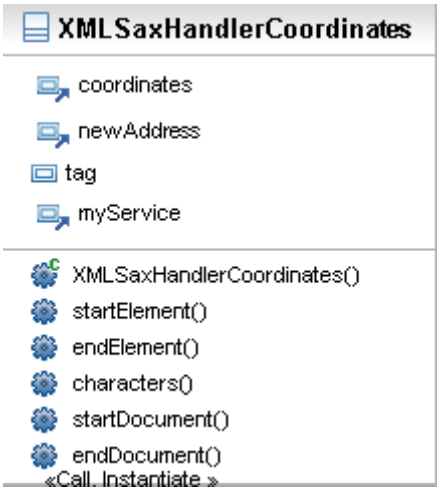


Figure 55 : Classe XMSaxHandlerCoordinates

Cette classe gère le parser SAX pour transformer le XML reçu par le serveur et contenant les points d'intérêt en une liste de Coordinates (coordonnées).
Comme pour le parser du jeton (XMLSaxHandlerToken), on gère tout simplement la stucture du XML reçu pour en retirer toutes les informations nécessaires à la construction d'une coordonnée.

Voici un exemple de XML contenant les points d'intérêts (désignées zone dans le XML) :

```
<?xml version="1.0" encoding="utf-8" ?>
<zones>
	<zone>
		<name>Maison</name>
		<latitude>46.7791238486708</latitude>
		<longitude>6.65638970436858</longitude>
		<dns>home.test.cc</dns>
		<addresses>
			<address>
				<mac>00:1A:8F:50:66:50</mac>
				<port>44555</port>
			</address>
		</addresses>
	</zone>
</zones>
```

7.2.2.4. NetworkProviderManager

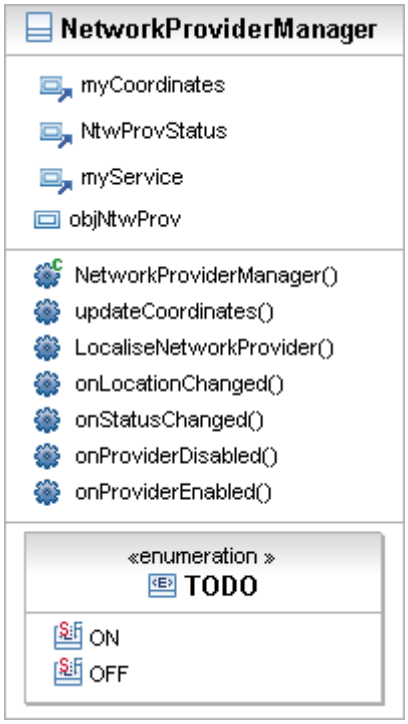


Figure 56 : Class NetworkProviderManager

Cette classe est celle qui va tout d'abord s'abonner à la réception de nouvelles coordonnées de position de l'utilisateur et ensuite les récupérer périodiquement.
Pour s'abonner, nous utilisons le code suivant : `requestLocationUpdates`
Ceci est fait grâce à un LocationManager qui nous est fourni par Android.
Une fois abonnés, nous recevons chaque mise à jour de position avec la fonction : onLocationChanged.
Pour pouvoir savoir quels points d'intérêts sont actifs, nous allons tout simplement les comparer à la nouvelle position et créer un tableau des indexes « actifs » correspondant aux positions des coordonnées actives.
Nous allons recevoir une coordonnée avec sa longitude, sa latitude et une certaine précision en mètres du réseau et nous allons regarder un par un, si un point d'intérêt est dans la zone de cette antenne.

7.2.2.5. ServiceCoordinatesReceiver

Figure 57 : Class ServiceCoordinatesReceiver

Comme nous l'avons vu précédemment, nous donnons la possibilité à l'utilisateur d'aller rechercher ses points d'intérêt sur le serveur à l'aide du bouton « Recup.Coord. » du menu.
Si le service est déjà démarré, il faudra du lui transmettre les nouvelle coordonnées depuis l'activité d'amorce.
Pour ce faire, nous utilisons un broadcastReceiver. Cette notion a été expliquée dans la partie technologies de ce rapport.

L'action qui est utilisée est : droidServiceFromDroiDynamics et les nouvelles coordonnées sont passées en extras à l'Intent du Broadcast.

7.2.2.6. StatusBarManager

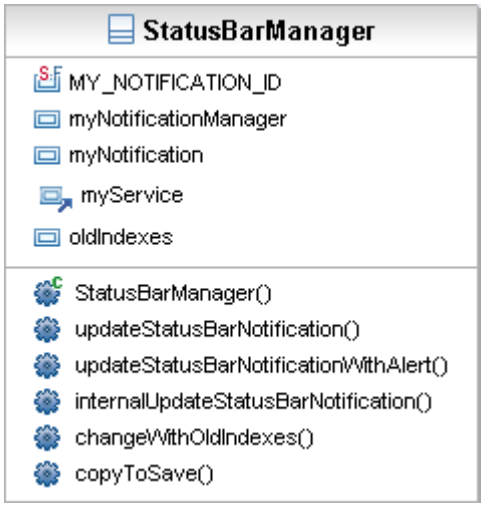


Figure 58 : Class StatusBarManager

Cette classe gère tout simplement les messages d'avertissement dans la barre des statuts du Smartphone.

Nous pouvons soit mettre à jour ce qui est affiché dans la barre des statuts avec la méthode updateStatusBarNotification ou alors avertir en même temps l'utilisateur avec un son, un témoin lumineux et le vibreur et ce avec la méthode updateStatutsBarNotificationWithAlert.

Si l'utilisateur appuie sur la notification, nous allons tout simplement créer une activité avec la classe LocationActionLayout.

Voici un exemple d'une notification dans la barre des statuts :

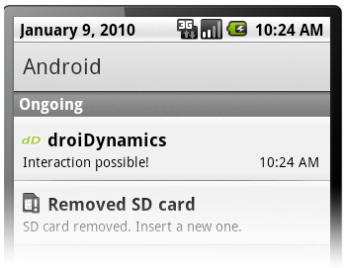


Figure 59 : Exemple d'alerte dans la barre de statuts

7.2.2.7. DroidService

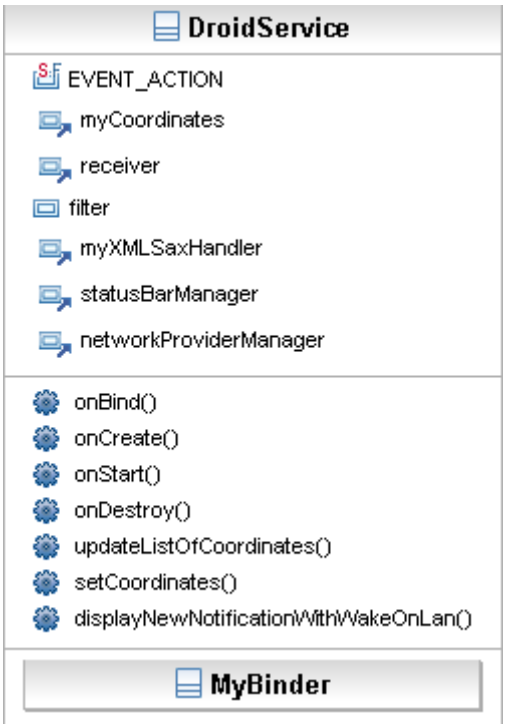


Figure 60 : Classe DroidService

Cette classe gère le service. Comme vu dans la partie « technologie », un service n'a besoin que des fonctions onCreate, onStart et onDestroy.

Nous pouvons tout simplement dire que le service va s'abonner aux changements de position de l'utilisateur via sa classe NeworkProviderManager et informer s'il y a des nouvelles interactions possibles via la classe StatusBarManager.

7.2.2.8. LocationActionLayout

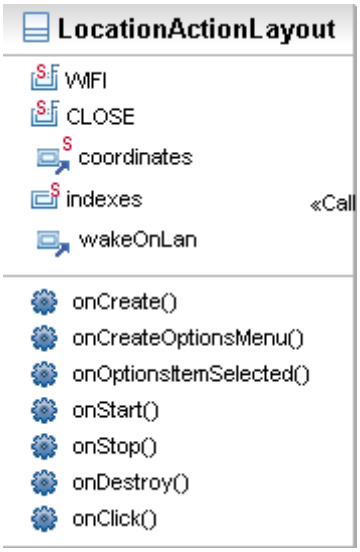


Figure 61 : Classe LocationActionLayout

Dans cette classe nous allons tout simplement gérer l'affichage de la fenêtre suivante :

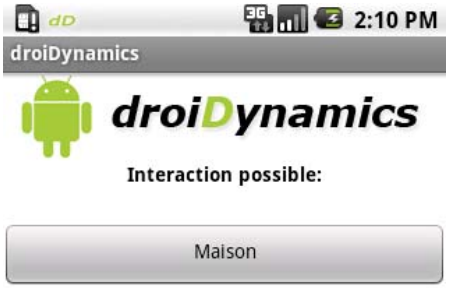


Figure 62 : Interface d'interaction

Nous donnons à l'utilisateur la liste de ses points d'intérêt à proximité de sa position géographique.
De plus lorsqu'il appuie sur le bouton, on va faire appel à la classe WakeOnLan pour gérer le réveil des ordinateurs.
Si l'utilisateur veut faire du Wake On Lan via Wifi, en appuyant sur la touche menu, il va accéder au paramétrage du Wifi sur le Smartphone afin de pouvoir l'allumer ou l'éteindre.

7.2.2.9. WakeOnLAn

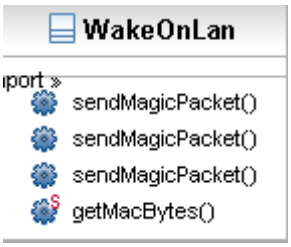


Figure 63 : Class WakeOnLan

Cette classe nous permet de gérer le Wake On Lan des différents ordinateurs. Elle se compose par trois surcharges de la fonction sendMagicPacket qui sont les suivantes :

1) On a besoin que de l'adresse Mac car on prend automatiquement l'IP 255.255.255.255 et le port 9
2) On a besoin de l'adresse Mac et le port. L'ip reste 255.255.255.255
3) On paramètre les trois attributs : Mac, Ip et port

Dans notre programme, nous utilisons la troisième surcharge car nous avons tous les paramètres nécessaires et que nous voulons permettre à l'utilisateur de faire du Wake On Lan via internet et via Wifi.

Pour plus d'informations concernant le code du programme, voir en annexe le code commenté.

8. Tests

8.1. Serveur Web

8.1.1. Site Web

Le site internet a été testé sur différents navigateurs récents. Nous n'avons pas tenu compte des anciennes versions de Microsoft Internet Explorer car la mise en place de scripts, en interaction avec Google Maps pour ces versions, demanderait un travail supplémentaire considérable. Le résultat est correct pour les navigateurs Mozilla Firefox, Google Chrome, Internet Explorer (dernière version) et Safari.

Chaque formulaire du site internet a été testé avec des valeurs vides et des valeurs problématiques (pour tester les injections SQL par exemple). Nous n'avons constaté aucun problème lors de ces tests, étant donné que toutes les entrées de formulaire sont contrôlées avant d'être traitées.

Nous avons également essayé toutes les parties du site web afin de passer par tous les codes qui ont été programmés. Encore une fois, aucun problème n'a été constaté lors de ces tests.

8.1.2. Site mobile

Pour le site mobile, les seuls tests à réaliser ont été d'effectuer des requêtes HTTP avec des paramètres particuliers ou manquants et de récupérer des résultats. Les résultats ont été une des erreurs expliquées plus haut dans le rapport. Donc nous gérons les erreurs expliquées à la page 38.

Comme pour les saisies utilisateur sur le site web, les paramètres passés sur le site mobile sont vérifiés avant leur utilisation. Le site vérifie d'abord s'ils existent, et si c'est le cas, ils sont contrôlés afin de ne pas interférer avec le reste du code.

Nous avons aussi essayé de recevoir un XML ne contenant aucune zone, ou des zones avec aucune adresse à réveiller, et cela n'a posé aucun problème, ni au niveau du site mobile pour la génération, ni pour leur compréhension par le téléphone mobile.

8.2. Android

Nous avons testé la consommation avec le GPS et le WIFI pour pouvoir déterminer s'il était utile de les utiliser ou non. Il s'est avéré que le GPS et le WIFI sont très gourmands en énergie. Ils consomment le double d'une utilisation normale de l'appareil, ce qui n'est pas acceptable pour une simple application de Wake On Lan.

De plus nous avons testé la précision et la rapidité du GPS. En endroit ouvert, le GPS met en moyenne 30 secondes pour capter un signal et nous envoyer des coordonnées précises (moins de 10 mètres de marge). Le problème est que si nous allumons le GPS dans un véhicule ou sous un toit, il a beaucoup de mal à trouver un signal et prend parfois plusieurs minutes à nous envoyer des coordonnées concernant notre position.

Des tests ont été menés avec des erreurs dans les champs de saisie des préférences :

- Si l'utilisateur a mal saisi son e-mail, le système lui relance l'activité de saisie et lui affiche en rouge un petit message :

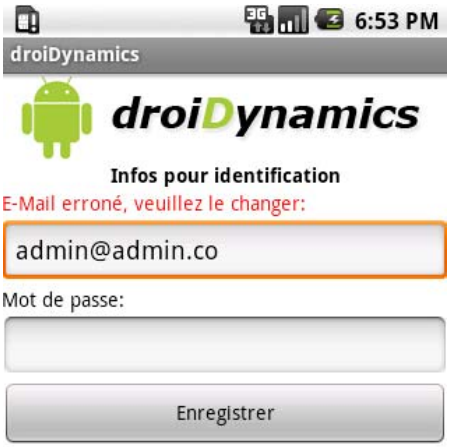


Figure 64 : Email erroné

- Si l'utilisateur saisi mal son mot de passe, idem que pour l'e-mail :

Figure 65 : Mot de passe erroné

- Si l'IMEI de l'appareil que l'utilisateur utilise n'est pas enregistré, nous affichons un message à l'utilisateur :

Figure 66 : Erreur : IMEI non enregistré

De plus, si la connexion internet n'est pas présente lorsque le Smartphone veut communiquer avec le serveur, nous affichons aussi un message à l'utilisateur :

Figure 67 : Erreur : pas de connexion possible

Pour finir, nous avons laissé tourner plusieurs jours notre application et avons essayé d'utiliser de différentes manières notre Smartphone. Nous avons constaté que lors d'une utilisation très intensive, notre téléphone détruit notre service car il a besoin de ressources et le relance ensuite. Cependant, cette coupure n'est que de très courte durée et n'est pas possible à calculer. Selon nos estimations, elle durerait moins d'une seconde.

9. Valorisation

Le programme sera valorisé lors de la présentation faite dans le cadre du cours. En-dehors du projet, il pourrait être mis en service et publié sur le market Android afin de le proposer en libre accès aux utilisateurs de Smartphones Android.

De plus, ce projet nous aura permis d'apprendre énormément sur Android et sur le hachage, ainsi que sur les notions liées à la sécurité entre les échanges d'informations via Internet.

10. Améliorations possibles

10.1. Serveur Web

10.1.1. Site Web

Le site Web peut être amélioré sur plusieurs axes. Le premier étant au niveau des fonctionnalités. Il serait bien sûr agréable d'avoir des fonctionnalités supplémentaires, tels que des possibilités de partage de zone entre plusieurs utilisateurs (pour une même famille par exemple).

Des changements peuvent également être apportées au niveau de l'ergonomie du site. Effectivement, certaines modifications permettraient à l'utilisateur de pouvoir gérer ses zones plus efficacement et plus rapidement. On peut notamment citer la page de modification d'une zone. Une carte pourrait s'afficher sur celle-ci afin de permettre à l'utilisateur de choisir de nouvelles coordonnées sur la carte sans avoir besoin de les saisir manuellement.

D'autres modifications peuvent être apportées au niveau des langues. Le site n'est disponible qu'en français. C'est suffisant dans le cadre du cours, mais pour une mise en production, le nombre d'utilisateurs possible s'en retrouverait fortement réduit.

On peut finalement noter que le design du site est très simple. Il pourrait être amélioré afin de rendre le projet plus attractif.

10.1.2. Site mobile

La principale amélioration réalisable sur le site mobile concerne bien sûr la réutilisation du code. Certaines parties de codes sont partagées entre le site mobile et le site complet, et étant donné que ce sont deux sites distincts, les fichiers sont dupliqués. Nous pourrions fusionner les deux sites. Il pourrait y avoir une détection automatique du navigateur (ou via un paramètre) afin de définir si l'on souhaite accéder au site mobile ou au site web depuis la même adresse. De cette façon, les fichiers utiles aux deux sites ne seraient plus dupliqués étant donné qu'il n'y aurait plus qu'une seule adresse d'accès.

10.2. Android

Concernant le programme Android, il y aurait quelques améliorations possibles :

- Mettre tout le code concernant l'activité d'amorce dans celui du service. De cette manière, le programme serait totalement transparent pour l'utilisateur. Notre programme ne tournerait donc plus qu'avec un service en tâche de fond et au lancement l'utilisateur n'aurait plus l'activité d'amorce.
  Cette modification peut être faite et testée en une journée de travail.
- Nous pourrions mettre un timer qui, chaque x temps, va chercher des mises à jour des points d'intérêt. Ceci lui permettrait d'être à jour concernant les modifications que peut faire l'utilisateur sur le site internet. Cependant, dans l'état actuel du programme, cette fonctionnalité est disponible manuellement via le menu de l'activité d'amorce.
  Cette modification peut être faite et testée en une demi-journée de travail.

11. Conclusion

Dans ce projet, nous avons touché à beaucoup de domaines et technologies différents. La difficulté était de les mettre ensemble et d'en sortir un résultat convaincant et utilisable.

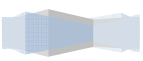
Le choix des technologies utilisées a été un élément clé dans la conception de nos programmes. Nous avons pris des directions précises afin de fournir une variété d'outils et d'options adéquats au cadre d'utilisation de notre projet.

Nous avons bien-entendu rencontré quelques problèmes et bugs tout au long du projet mais nous avons réussi à les combler et à continuer notre programmation en respectant les délais qui nous étaient imposés. Ceci étant dit, nous avons dressé la liste des améliorations possibles à apporter à nos programmes.

Notre interface a été conçue pour être la plus intuitive possible, ergonomique et agréable à utiliser. La partie Android est complètement compatible avec une utilisation tactile. Nous pouvons donc naviguer facilement du bout des doigts dans notre logiciel.

Nous pouvons dire qu'après avoir réalisé un projet du début à la fin en tenant compte des divers imprévus et contraintes, nous avons acquis une expérience supplémentaire et indispensable pour notre projet de Bachelor et à plus long terme pour les prochains projets que nous ferons en entreprise.

Carlo Criniti et Claudio Pimpao



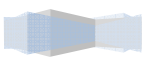
12. Bibliographie

12.1. Serveur Web

- Référence PHP : http://www.php.net
- Google Maps API 3 Référence : http://code.google.com/apis/maps/documentation/v3/
- Référence HTML : http://www.w3schools.com/html/default.asp
- Référence CSS : http://www.w3schools.com/css/default.asp
- Référence Javascript : http://www.w3schools.com/js/default.asp
- Tutoriaux et astuces CSS : http://www.alsacreations.com/

12.2. Android

- Android – Développer des applications mobiles pour les Google Phones, Edition Dunod, Paris, 2009, ISBN 978-2-10-053181-3
- Site officiel Android : http://www.android.com
- Site du développeur Android officiel : http://developer.android.com
- http://www.openhandsetalliance.com
- http://fr.wikipedia.org/wiki/Android



13. Table des illustrations

Figure 1 : Icône de l'application 5
Figure 3 : Image de notre interface de login 6
Figure 4 : Image de notre application tournant sur un émulateur du Smartphone HTC Magic 6
Figure 2 : Image de notre interface d'ajout de point d'intérêt 6
Figure 5 : Vue Satellite de Yverdon-les-Bains suite à une recherche sur Google Maps 17
Figure 6 : Vue en mode carte de Yverdon-les-Bains suite à une recherche sur Google Maps 17
Figure 7 : Infowindow affichée sur la page d'accueil du site Web 18
Figure 8 : Pourcentage de consommateurs qui planifient d'acheter un Smartphone Android dans les prochains 90 jours 19
Figure 9 : Progression plus importante pour Android que pour ses concurrents 20
Figure 10 : Architecture Android 21
Figure 11 : Notre environnement de développement sur Eclipse Galileo avec le SDK d'Android et l'émulateur 22
Figure 12 : Ajout d'un nouveau projet sur Eclipse utilisant le SDK Android 23
Figure 13 : Structure du projet 24
Figure 14 : Différents états d'une Activity 25
Figure 15 : Différentes étapes dans la vie d'un service 27
Figure 16 : Nexus One de Google construit par le fabriquant de Smartphones HTC 30
Figure 17 : Ecran d'accueil du site Web 31
Figure 18 : Ecran de création d'un compte sur le site Web 32
Figure 19 : Infowindow d'ajout de zone 33
Figure 20 : Page de modification d'une zone 33
Figure 21 : Liste des ordinateurs de la zone 34
Figure 22 : Page de modification des paramètres d'accès 34
Figure 23 : Procédure de chiffrement d'une donnée sensible 36
Figure 24 : Ecran d'accueil du site Web 37
Figure 25 : Diagramme de représentation générale des Activités et Service du programme 41
Figure 26 : Icône dD de la barre de statuts 42
Figure 27 : Notification d'interaction possible 42
Figure 28 : Exemple d'interaction avec un point d'intérêt 42
Figure 29 : Fonctionnement de l'activité d'amorce 43
Figure 30 : Les différentes vues de notre programme 45
Figure 31 : Diagramme de classes, partie activité d'amorce 52
Figure 32 : Classe Hashing 53
Figure 33 : Classe dDException 53
Figure 34 : Classe Preferences 53
Figure 35 : Classe XMLSacHandlerToken 54
Figure 36 : Classe ClientHTTP 54
Figure 37 : Génération de la demande des zones 54
Figure 38 : Classe UserNotification 55
Figure 39 : Toast 55
Figure 40 : Boîte de dialogue 55
Figure 41 : Classe PrefLayout 56
Figure 42 : Interface de saisie des préférences 57

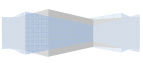


Figure 43 : Email erroné 58
Figure 44 : Mot de passe erroné 58
Figure 45 : Classe MainScreenLayout 58
Figure 46 : Layout principal sans erreur 58
Figure 47 : Layout principal avec exemple d'erreurs 58
Figure 48 : Menu de l'activité d'amorce 59
Figure 49 : Dialogue d'informations 59
Figure 50 : Classe AmorceActivity 59
Figure 51 : Classe Controller 60
Figure 52 : Diagramme de classes, partie Service 61
Figure 53 : Classe Address 62
Figure 54 : Classe Coordinate 62
Figure 55 : Classe XMSaxHandlerCoordinates 63
Figure 56 : Class NetworkProviderManager 64
Figure 57 : Class ServiceCoordinatesReceiver 64
Figure 58 : Class StatusBarManager 65
Figure 59 : Exemple d'alerte dans la barre de statuts 65
Figure 60 : Classe DroidService 65
Figure 61 : Classe LocationActionLayout 66
Figure 62 : Interface d'interaction 66
Figure 63 : Class WakeOnLan 66
Figure 64 : Email erroné 68
Figure 65 : Mot de passe erroné 68
Figure 66 : Erreur : IMEI non enregistré 68
Figure 67 : Erreur : pas de connexion possible 69